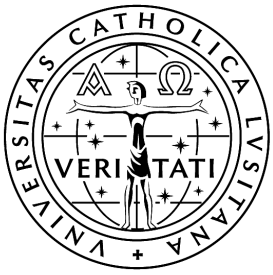


UNIVERSIDADE CATÓLICA PORTUGUESA

MODELLING THE LIVE-ELECTRONICS IN ELECTROACOUSTIC MUSIC USING PARTICLE SYSTEMS

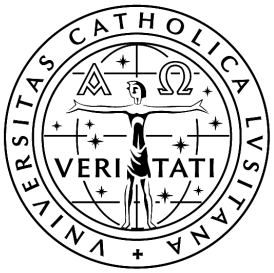
Dissertation submitted to the Portuguese Catholic University in partial fulfillment of requirements of the Doctoral Degree in Science and Technologies of the Arts – Computer Music

by

André Venturoti Perrotta

ESCOLA DAS ARTES

Abril 2015



UNIVERSIDADE CATÓLICA PORTUGUESA

MODELLING THE LIVE-ELECTRONICS IN ELECTROACOUSTIC MUSIC USING PARTICLE SYSTEMS

Dissertation submitted to the Portuguese Catholic University in partial fulfillment of requirements of the Doctoral Degree in Science and Technologies of the Arts – Computer Music

By André Venturoti Perrotta

Supervised by Professor Luis Gustavo Pereira Marques Martins

ESCOLA DAS ARTES

Abril 2015

Acknowledgements

The work that I present in this dissertation started a long time ago, many years before I've even imagined becoming a Ph.D candidate. Along the winding path the led me here, difficult challenges had be to overcome and critical life changing choices had to be made. I did not have to walk down this tortuous path alone. At every step and at every problem or dilemma, I could always seek help and guidance from my family, friends and masters.

This thesis is dedicated to all of those who directly or indirectly participated in this journey.

To my parents Rosangela and José Augusto, who guided my whole life towards the search for knowledge and education and who always gifted me with unconditional support for my ideas and choices.

To my whole family, who enlightened me with care and culture.

To my close friends from *Bonfa*, Caio, Vitinho, Thiago and Jun, for filling my life with joy and happiness.

To my dear friends from Portugal, Vasco, João and André, who took me in as family and helped relief the hard times of the immigrant life.

To my masters, Sergio Luis Morelhão, for opening the door for science and logical thinking, and Ulisses Rocha for helping me to find my musical sensibility and artistic character.

To my friend and professor Flo Menezes, for absolutely transmitting all his knowledge and offering me the opportunity to prove myself and work amid distinguished composers and musician of the contemporary electroacoustic music scene.

To my advisor Luis Gustavo Martins, for guiding me throughout this research with stunning precision.

To my beloved wife Raquel, to whom I dedicate not only this thesis, but our entire life and future.

Abstract

Contemporary music is largely influenced by technology. Empowered by the current available tools and resources, composers have the possibility to not only compose with sounds, but also to compose the sounds themselves.

Personal computers powered with intuitive and interactive audio applications and development tools allow the creation of a vast range of real-time manipulation of live instrumental input and also real-time generation of sound through synthesis techniques. Consequently, achieving a desired sonority and interaction between the electronic and acoustic sounds in real-time, deeply rely on the choice and technical implementation of the audio processes and logical structures that will perform the electronic part of the composition.

Due to the artistic and technical complexity of the development and implementation of such a complex artistic work, a very common strategy historically adopted by composers is to develop the composition in collaboration with a technology expert, which in this context is known as a *musical assistant*. In this perspective, the work of the musical assistant can be considered as one of translating musical, artistic and aesthetic concepts into mathematical algorithms and audio processes.

The work presented in this dissertation addresses the problem of choosing, combining and manipulating the audio processes and logical structures that take place on the live-electronics (i.e the electronic part of a mixed music composition) of a contemporary electroacoustic music composition, by using particle systems to model and simulate the dynamic behaviors that reflect the conceptual and aesthetic principles envisaged by the composer for a determined musical piece.

The presented research work initiates with a thorough identification and analysis of the

agents, processes and structures that are present in the live-electronics system of a mixed music composition. From this analysis a logical formalization of a typical live-electronics system is proposed, and then adapted to integrate a particle-based modelling strategy.

From the formalization, a theoretical and practical framework for developing and implementing live-electronics systems for mixed music compositions using particle systems is proposed. The framework is experimented and validated in the development of distinct mixed music compositions by distinct composers, in real professional context.

From the analysis of the case studies and the logical formalization, and the feedback given by the composers, it is possible to conclude that the proposed particle systems modelling method proves to be effective in the task of assisting the conceptual translation of musical and aesthetic ideas into implementable audio processing software.

Resumo

A música contemporânea é amplamente influenciada pela tecnologia. Os recursos tecnológicos atualmente disponíveis permitem que os compositores criem com sons e ao mesmo tempo criem os sons em si próprios.

Os atuais aplicativos e ferramentas de software focados no desenvolvimento, controle e manipulação de processamentos de áudio, permitem a elaboração de diversos tipos de tratamentos e sínteses de som com a capacidade de serem executados e manipulados em tempo real. Consequentemente, a escolha dos algoritmos de processamento de áudio e suas respectivas implementações técnicas em forma de software, são determinantes para que a sonoridade desejada seja atingida, e para que o resultado sonoro satisfaça os objetivos estéticos e conceituais da relação entre as fontes sonoras acústicas e os sons eletrônicos presentes em uma composição eletroacústica de caráter misto.

Devido à complexidade artística e técnica do desenvolvimento e implementação do sistema de eletrônica em tempo real de uma composição eletroacústica mista, uma estratégia historicamente adotada por compositores é a de desenvolver a composição em colaboração com um especialista em tecnologia, que neste contexto é usualmente referido como *assistente musical*. Nesta perspectiva, o trabalho do assistente musical pode ser interpretado como o de traduzir conceitos musicais, artísticos e estéticos em algoritmos matemáticos e processamento de áudio.

O trabalho apresentado nesta dissertação aborda a problemática da escolha, combinação e manipulação dos processamentos de áudio e estruturas lógicas presentes no sistema de eletrônica em tempo real de uma composição de música eletroacústica contemporânea, e propõem o uso de sistemas de partículas para modelar e simular os comportamentos dinâmicos e morfológicos que refletem os princípios conceituais e estéticos previstos pelo compositor para uma determinada composição.

A parte inicial do trabalho apresentado consiste na identificação e análise detalhada dos agentes, estruturas e processos envolvidos na realização e execução do sistema de eletrônica em tempo real. A partir desta análise é proposta uma formalização lógica e genérica de um sistema de eletrônica em tempo real. Em seguida, esta formalização é modificada e adaptada

para integrar uma estratégia de modelagem através de sistemas de partículas.

Em sequencia da formalização lógica, um método teórico e prático para o desenvolvimento de sistemas de eletrônica em tempo real para composições de música mista é proposto. O teste e consequente validação do método se dá através de sua utilização na realização da eletrônica em tempo real para obras de diferentes compositores.

A análise dos casos de estudo e da formalização lógica, e também o parecer e opinião dos compositores, permitem concluir que o método proposto é de fato eficaz na tarefa de auxiliar o processo de tradução dos conceitos musicais e estéticos propostos pelos compositores em forma de algoritmos e processamentos de som implementados em software.

Contents

List of Figures

List of Tables

Nomeclature

Acronyms / Abbreviations

ADC Analog to digital converter

CPU Central processing unit

GPU Graphical processing unity

GUI Graphical user interface

MA Musical assistant

MIDI Musical Instrument Digital Interface

MIR Music information retrieval

MVC Model-View-Controller software architecture

Chapter 1

Introduction

1.1 Context and Motivation

The work presented in this dissertation lies on the general field of *Computer Music*, which interfaces music technology with music composition. More specifically, this dissertation is framed in the universe of contemporary electroacoustic music, and it focuses on the specific genre of mixed music, which congregates instrumental writing with electronic devices and resources. In this context, the electronic part of a mixed music composition is commonly known as the *live-electronics*, and its technical implementation as *live-electronics system*.

Contemporary music is largely influenced by technology. With the current available tools and resources, composers have the possibility to compose the music and the instruments, or, as stated by Jean-Claude Risset:

> "... composing with sounds and composing the sounds themselves ..." (Risset, 1978)

Personal computers powered with intuitive and interactive audio applications and development tools allow the creation of a vast range of real-time manipulation of live instrumental input and also real-time generation of sound through synthesis techniques. Consequently, achieving a desired sonority and interaction between the electronic and acoustic sounds in real-time, deeply rely on the choice and technical implementation of the audio processes

and logical structures that will perform the electronic part of the composition.

Due to the artistic and technical complexity of the development and implementation of such a complex artistic work, a very common strategy historically adopted by composers is to develop his/her composition in collaboration with a technology expert, which in this context is known as a *musical assistant*. In this perspective, the work of the musical assistant can be considered as one of translating musical, artistic and aesthetic concepts into mathematical algorithms and audio processes.

A composer that is proficient in computer programming, can take advantage of the available tools and create audio applications that relate closely with his/her own musical idea. However, on the other hand, a professional musical assistant must be able to work with different composers and conceptually different projects.

Despite the fact that the topics related to musical composition, aesthetics, musical informatics and audio processing are very well-established in the academic and commercial worlds, the work and practice of a musical assistant is not yet formally developed in terms of formalized theory and practice methodologies.

The work presented in this dissertation attempts to address the problem of choosing, combining and manipulating the audio processes and logical structures that take place on the live-electronics (i.e the electronic part of a mixed music composition) of a contemporary electroacoustic music composition, by using particle systems to model and simulate the dynamic behaviors that reflect the conceptual and aesthetic principles envisaged by the composer for a determined musical piece. The method presented is inspired by computer programming methodologies, parameter mapping strategies, scientific modelling techniques and musical composition concepts.

Even though the technical development of a live-electronics system must be deeply related to the artistic concepts of the respective musical composition, the relationship between electronic and acoustic sources, from a musical composition perspective, is out of the scope of this thesis.

1.2 Thesis Statement

This thesis tries to answer the following research question:

> "Is it possible to use logical constructions and computational methods to mediate the process of translating abstract artistic ideas into executable software in mixed music works, using for that purpose an implementation method based on particle systems ?"

1.2.1 Main Objectives

The main objective of this research is to develop a theoretical and practical framework that can be used by composers and musical assistants on the technical implementation of live-electronics systems.

Hence, it is possible to enunciate the research objectives as follows:

- To develop a logical formalization of a live-electronics system.
- To develop a theoretical and practical framework for modelling and implementing live-electronics systems using particle systems.
- To apply the developed methods and tools in the implementation of the live-electronics of mixed music compositions.

1.2.2 Current State

Current live-electronics systems present a complex combination of techniques and aesthetics. The offer of well-established and affordable dedicated software tools and programming environments such as Max[1], Pd[2] and Supercollider[3] enables a proficient programmer to

[1] www.cycling74.com
[2] www.puredata.info
[3] www.supercollider.sourceforge.net

arrange, create and modify a considerable variety of distinct digital audio processing algorithms such as pre-composed sounds, real time synthesis, real time audio effects and real time spatialization.

In 1999, Jean-Claude Risset (Risset, 1999) wrote about the limitations of real-time computer music applications and composition. He directed his remarks towards the technology limitations such as the inability of processing complex sounds through layering of several distinct voices, and also towards the methodology involved in implementing and composing the sounds and sonorities of a live-electronics system. In what regards the technological issues, it is safe to say that the currently available and affordable software and hardware tools are able to fulfill almost any creative ambition (Tremblay, 2006). However, the methodological issues are still present.

One of the methodological problems pointed out by Risset (Risset, 1999), stated that the difficulty in controlling multiple parameters of an audio process leads to an empirical strategy of random trials and errors, and consequently, the result seldom reflects the initial objective imagined by the composer. This problem still persists, and in fact, due to the increased complexity of the current live-electronics systems and consequently the increased number of control parameters that are needed to be manipulated in real-time in order shape the final sound, the problem is even more evidenced.

Problems related to the control of multiple parameters of audio processes in real-time have been widely researched under the realm of parameter mapping. In that matter, several solutions have been proposed with implementation techniques ranging from as simple as one-to-one variable mapping with a deterministic linear function, to many-to-many mapping with stochastic and chaotic distribution (Hunt and Wanderley, 2002; Nort et al., 2014).

Phillipe Manoury (Manoury, 1998) has also approached the challenges of developing and performing mixed music. In his work, he discusses the possibilities of synchronization of the electronic and instrumental parts, and how to compose and implement live-electronic systems that presents a tight relationship between the electronic sounds and the instrumental interpretation.

The process of choosing and combining the audio processes that take place on a live-

electronics system has been widely studied on a musicological and compositional perspective. Mike Frengel (Frengel, 2010) proposed a multidimensional framework to analyze the possible relationships between the acoustic and electronic sources from a musicological and perceptual point of view. Flo Menezes (Menezes, 2002) discusses the interaction between instrumental and electronic sources from an aesthetic perspective and determines a continuum that goes from maximum *fusion* to maximum *contrast*.

On the technical implementation perspective, or as to say, the musical assistant perspective, there is no methodology dedicated to the process of choosing and combining audio processes derived from conceptual and musical ideas (what we denote as conceptual mapping). Studies and resources on this topic tends to represent the solutions developed for a specific musical piece (Tremblay, 2006) or that relate with the methods of a specific composer (Hoffmann, 2002).

Particle systems are widely used in several research areas for modelling and simulating complex events such as physical, natural, chemical phenomena, etc. Essentially a particle system is an useful modelling technique for describing systems that present highly dynamic and stochastic behaviors. In computer music, particle systems have been used for data sonification, physically based synthesis techniques, algorithmic composition and audio effects (Blackwell and Young, 2004; Cádiz and Kendall, 2005; De Poli and Rocchesso, 1998; Sturm, 2000). Nevertheless, the use of particle systems on a structural level on the implementation of live-electronics software has not been widely researched. In that matter, the project *Interactive Swarm Orchestra* at the ICST Zurich, has explored a specific class of dynamic behavior (swarm behavior) and developed a set of particle system based software tools for creating sound synthesis and spatialization using swarm algorithms (Bisig et al., 2008).

Even though several research and artistic projects have been developed using particle systems as an effective way for modelling and simulating complex sonic structures, there is still a long way before particle system based methods and tools reaches the same maturity level in the computer music domain, as it has reached in the computer graphics and physical computing domains.

1.3 Research Methodology

This research work is a direct consequence of the authors professional work experience as a musical assistant and technology director for several artistic projects. In that sense, the research methodology applied on the development of this work can be considered as practice-based and practice-led, as interpreted by Haseman (Haseman, 2006). Where in this context, the terms practice-based and practice-led refers to the process of extracting the research questions and proposed solutions from the practical experience. Furthermore, the developed frameworks are also validated by applying them in practice. The "original contribution to knowledge" is not only the proposed theoretical framework, but also the discussion of their application in real artistic projects.

Despite the fact that research questions and objectives are a consequence of pragmatical developments, the research also presents a theoretical component. In order to focus the research work and enunciate a concise research question, a thorough analysis of the work from the most relevant authors on the topics related to computer music techniques and electroacoustic music composition have been carried on. The analysis observed textual publications, live-electronics scores, recordings and software source code.

Furthermore, in order to propose a solution for the given research question, a theoretical logical formalization of a general live-electronics system has been developed. From this formalization it is possible to understand the research question from a technical perspective and also to predict the impact of the proposed solution.

The validation of the proposed solution is extracted both from the logical formalization and through its practical application in the development of mixed music works with distinct composers. The validation criteria is subjective in its form, meaning that no metric parameter is applied to evaluate the performance of the composition. The evaluation is presented as a discussion of the implications of the proposed method in each work.

Chapter 2

Live-electronics

2.1 Background

As many other terms associated with the music of the 20th century, electroacoustic music refers to the means and methods of music production as well as to a particular aesthetic and compositional approach. Electroacoustic music is defined as the genre of music in which electronic means are used as an intrinsic part of the compositional and performative processes. Thereupon, electronic tools and technology is used in the manipulation and control of sound material as well as in the composition and generation of the sound material itself.

An electroacoustic music concert can be of two different formats: *Acousmatic*, a term supposedly original from the philosopher Pythagoras, used to designate his method of teaching his students behind a curtain so that his physical presence and sight would not disturb their attention, which in this context means that the music is performed exclusively by fixed electronic medium (pre-produced recorded sounds) via loudspeakers, with no other visual or physical presence on stage; and *live electronic music*, where the electronic sounds are performed in real-time with or without additional acoustic instruments (Peignot, 1960; Sadie, 1980).

The conceptual meaning of the term and its associated musical aesthetic is however more complex and depends on the historical context. In the scope of this research, the

definition of electroacoustic music is framed inside the erudite music context that emerged as a direct consequence of the combination of new technological advances, such as recording equipment and electronic sound generators, with new western music composition paradigms from the beginning of the 20th century.

In its beginning, in the 1950s, electroacoustic music had two main aesthetic expressions: *Concrete Music* (in french: *Musique concrète*) and *Electronic music* (in german: *Elektronische Musik*). The former was created in 1948 by Pierre Schaeffer, who at that time worked as sound engineer for the *Radiodiffusion française* in Paris. The term coined by Schaeffer referred to the music that was only possible due to the advent of the tape recorder, which provided composers with the means for creating musical compositions by recording, manipulating and organizing acoustic sounds from any kind of source, hence fulfilling John Cage's prediction that the electrical instruments (or means) would

> "Make available for musical purposes any and all sounds that can be heard."(Cage, 2011)

The most significant works of this genre were developed at the *Groupe de Recherche de Musique Concrète*, founded by Schaeffer and Pierre Henry (who also worked as sound engineer together with Schaeffer) in 1950. The later originated at the studio of the *Radio Cologne* in Germany in 1951, referred to music composed purely of synthesized sounds, made possible by the advent of audio generators (oscillators) that could produce waveforms with programmed frequencies. The most significant works of this genre were composed by Karlheinz Stockhausen, and his first pieces *Studie I* and *Studie II* are considered iconic symbols of this period.

Despite the conceptual differences and even rivalry between these two branches of electroacoustic music, they shared a common characteristic: both used a fixed medium for delivering the final musical composition. The first electroacoustic musical pieces were mainly *Acousmatic*, which in this context means that a concert or performance is carried on by loudspeakers, there is no live instrumental interaction with the electronic medium.

Experiments using electric instruments in live performances have been done since the beginning of the 20th century inspired by the inventions of new instruments such as the

Thaddeus Cahill's *thelharmonium* (1906), Lev Termen's *theremin* (1920) and Maurice Martenot's *ondes martenot* (1928). However, due to the limited quality and malleability of the sound generated by these instruments, they did not succeed in establishing a new musical genre. Nevertheless, it served as great inspiration for the development of the subsequent electronic devices and John Cage's *Imaginary Landscape No. 1* (1939) for two variable-speed turntables, frequency recordings, muted piano, and cymbal is regarded as the first live electronic music composition.

It was not long after the birth of electroacoustic music in the late 1940s that composers started to realize the potential of combining pre-recorded/composed fixed medium sounds with live instruments and consequently closing the gap between divergent musical aesthetics proposed by the concrete and electronic music. The pioneer works of this kind were *Musica su due dimensioni I* (1953) for flute, cymbal and tape by Bruno Maderna and *Orphée* 53 (1953) for soprano and tape by Pierre Schaeffer and Pierre Henry. This music that blends electronic means with live acoustic instruments is referred as *mixed music*. In this genre, Stockhausen's Kontakte (1960) for piano, percussion and tape is regarded as a breaking point of electroacoustic music and marks a new path of composition paradigms and aesthetics concepts.

The next step in electroacoustic music was the integration of electronic manipulation of live acoustic sources. This came as a way to better integrate electronic and acoustic sounds in a performance. The use of a fixed medium such as taped music in performances represented a restriction for the organic integration of a live acoustic performer. Pioneers such as Stockhausen and Cage attacked this issue by composing pieces that require manipulation of microphones and other electronic devices for sound transformation and generation in real-time. Cage's *Cartridge Music* (1960) and Stockhausen's *Mikrophonie I* (1964) are references of this period.

In 1957 a new revolution in music was starting to rise. In Murray Hill, New Jersey - USA, a team of engineers at the Bell Telephone Laboratories developed the first computer generated sounds. Among the researches was Max Mathews, who immediately associated that technology with music. He developed the first ever computer program dedicated to

generating sound, the *Music I*, which although very simple and restrictive, was the first step towards our current music technology state-of-art. The *Music I* software and the subsequent *Music N* software family, anticipated several computer programming paradigms and strategies that are used today.

As the years went by, the initial limitations such as cost and processing power that restricted the use of computers in music became less significant. Advances in computer technology were followed by advances in audio processing and synthesis techniques. Art and science evolved in a feedback and feedforward loop in such a way that it is now impossible to dissociate music from technology. This music is now referred as *computer music*. (Chadabe, 1997; Morgan, 1991; Sadie, 1980; Taruskin, 2009)

As computers become ubiquitous, the meaning of the term *computer music* became less significant and it is now possible to say that computer music is a misleading term, thus any kind of music involves computers at some point of its production pipeline from the composer to the audience ears. As a consequence several distinct genres of music with completely different aesthetics, styles and purposes have appeared. The concepts, aesthetics and technology of the pioneers are now dissolved and permeate all musical paths, from the most commercial and focused on the entertainment to the most radically experimental focused only in the artistic core of music.

2.2 Mixed Music

The term *mixed music*, as it first appeared in the early 1950s, was used to describe compositions that merged instrumental music and electronic sounds (as presented in section 2.1). In that time, the electronic sounds of a mixed music composition had to be elaborated in studio and recorded into fixed support (such as magnetic tape). A performance of mixed music involved "live" instruments played by real musicians on stage and the playback of the electronic sounds previously recorded into the fixed support medium. In other words, a mixed music composition represented the merge of instrumental and acousmatic music, and accordingly, instrumental and acousmatic composition (Frengel, 2010).

From its early years until the present day, many questions related to the confrontation and integration of instrumental and electronic sounds have been raised by composers and computer music scientists. Questions such as, how to create and transpose the instrumental interpretative possibilities to the electronic part? How to create a natural flow between synchronicity and non-synchronicity? This questions are a consequence to the fact that electronic music is very young in comparison to its instrumental counterpart. Also, due to the fact that it is not possible to project the matured and established composition principles of instrumental music onto the electronic music domain. Composing sounds as opposed to composing with sounds, are two different paradigms that cohabit the same space, but nevertheless of completely distinct nature (Menezes, 2002).

Mixed music is currently an important genre of contemporary erudite music. Important music festivals such as the Warsaw Autumn, Darmstadt Ferienkurse für neue musik, Ultraschall Berlin, Tage für neue musik Zürich, among others, present mixed music compositions as an important part of their concerts program, featuring established and historical pieces as well as new commissioned works. Renowned ensembles and orchestras of contemporary music such as the Arditti Quartet, Ensemble Intercontemporain, Linea Ensemble, Remix Ensemble, Tonhalle Orchestra Zürich, among others, routinely perform mixed music works.

Current established composer such as Flo Menezes, Philipe Manoury, Marco Stroppa, among others, are part of a generation that have started and matured their music alongside the time computer music started to become accessible and affordable (1980s), thus mixed music is a solid and significant part of their work.

2.2.1 Composition and Musical Writing

The act of writing music transcends the pragmatical aspects of communication between composer and musician, and that of musicological documentation. It is an act of creation itself; it is the process of translating creative musical ideas into performable matter. Therefrom, as in other genres of erudite music prior and contemporary to electroacoustic music, the composition process of a mixed music piece is deeply dependent on musical writing.

However, the process of translating the creative musical ideas into performable matter differs drastically in the instrumental and electronic domains. Whilst the instrumental composition and creative process is constructed by means of writing music using the traditional notation (common practice notation), there is no standard symbolic language capable of expressing the electronic part. Electronic sounds can be represented graphically, mathematically, by programming language or other form of protocol. Usually this acts as a *post–factum* representation of a sound that is already conceived and technically implemented; it is an explanation of what we've heard, not of what we wish to hear.

Instrumental writing in the 20th century music already presented itself as a very complex task due to the fact that the traditional music notation didn't predict the non-metrical and non-tonal forms. However, traditional music notation still thrives as a connection medium between the composer and the instrumental performer (and conductor). It allows the composer to express the most substantial musical structures and at the same time leaves room for the performer to contribute with his understanding and personal interpretation of the work (Manoury, 1998).

Despite its merits, the common practice notation can not be employed in the creative and compositional processes of elaborating electronic sounds. Composing the sounds themselves, as opposite to composing with sounds, implies that many other aspects beyond pitch, duration and overall dynamics must be controlled to the finest grain. The precise description of a timbre and its dynamic morphology throughout infinitesimal and large time steps is far beyond the reach of the traditional writing (Nicolas, 2002).

In conclusion, in instrumental music the composer designs musical structures, timbre and dynamics by writing notes and figures in paper. In electronic music the composer designs musical structures, timbre and dynamics by manipulating recorded sounds and/or generating new ones with electronic means.

2.2.2 Real-time Implications

In the beginning of electroacoustic music in the late 1940s until the late 1970s, composing for instruments and composing electronic sounds were invariably deferred-time activities.

The process of composition took place before the music or sounds were performed. Hence, composition and performance were situated apart in time. In instrumental music, this was an intrinsic characteristic, but in electronic music this was an imposition of the technical limitations.

The technical restrictions of that time are long gone, and computers are now able to perform all kinds of audio processes in real-time. On a pure technical perspective a real-time computer process means that there is no perceivable delay between a command and the response output of an audio process (Roads and Strawn, 1985). On a musical perspective this opened up wide range of possibilities that affects all levels of the music creation and performance. Controlling the timbre and dynamics of sound in real-time represents the possibility of shaping the sound as if it was truly palpable (at least in theory), and thus the process of achieving complex and expressive timbres would become more intuitive and generous.

The possibility of controlling parameters of an audio process software and hearing the results right away is something incredibly remarkable in regards to computer software and hardware. However, this potential also raises very important questions in regards to its applications in music composition. As stated by Jean-Claude Risset:

> "... real-time operation is in fact better suited to performance and improvisation than to genuine composition. Composition is not – or should not be – a real-time process. ..." (Risset, 1999).

Live electronic music represented the way to inject interpretative potential to the electronic music (which in that time used to rely on fixed medium). And this is exactly the potential that real-time computer processing enables. Not surprisingly, the term live electronic music is currently interchangeable with the term real-time computer music, or interactive computer music.

2.2.3 Agents and Processes

The process of creating a mixed music composition, from the composers ideas and concepts, to a performance involving musicians, acoustic instruments and electronic sounds, is a process that requires a combination of music composition, scientific and artistic research, performance practice and creative engineering.

Figure 2.1 describes the flow of information that usually takes place in the development of a mixed music project, as well as the involved agents and processes. This scheme does not consider the possible retroactive interaction between electronics and musical writing itself, as described so pertinently by the composer Philippe Manoury as “partitions virtuelles" (virtual scores)(Manoury, 1998), since it surpasses the goal of our purposes in this Thesis. It does not consider neither the number of performers nor the fact that not all the performers are necessarily submitted to the live-electronics processes in a mixed work with electronics in real-time. The right side of the figure shows the development workflow and on the left side the performance workflow. The blocks inside the grey rectangle represent the structures that need to be implemented by the musical assistant.

2.2.3.1 The Composer

The composer is the principal (if not unique) author of the artistic musical work. He is responsible for setting the musical objectives and projecting the overall structure and aesthetic of the work.

2.2.3.2 The Performers

The performers are responsible for playing the finished musical work. The musicians and conductor engage with instrumental part of the music and the live-electronics performer (also known as digital performer) is responsible for controlling the overall flow of the live-electronics.

An important part of their work is to balance their personal intuitive understanding and consequent interpretation of the composed music with the intended interpretation idealized

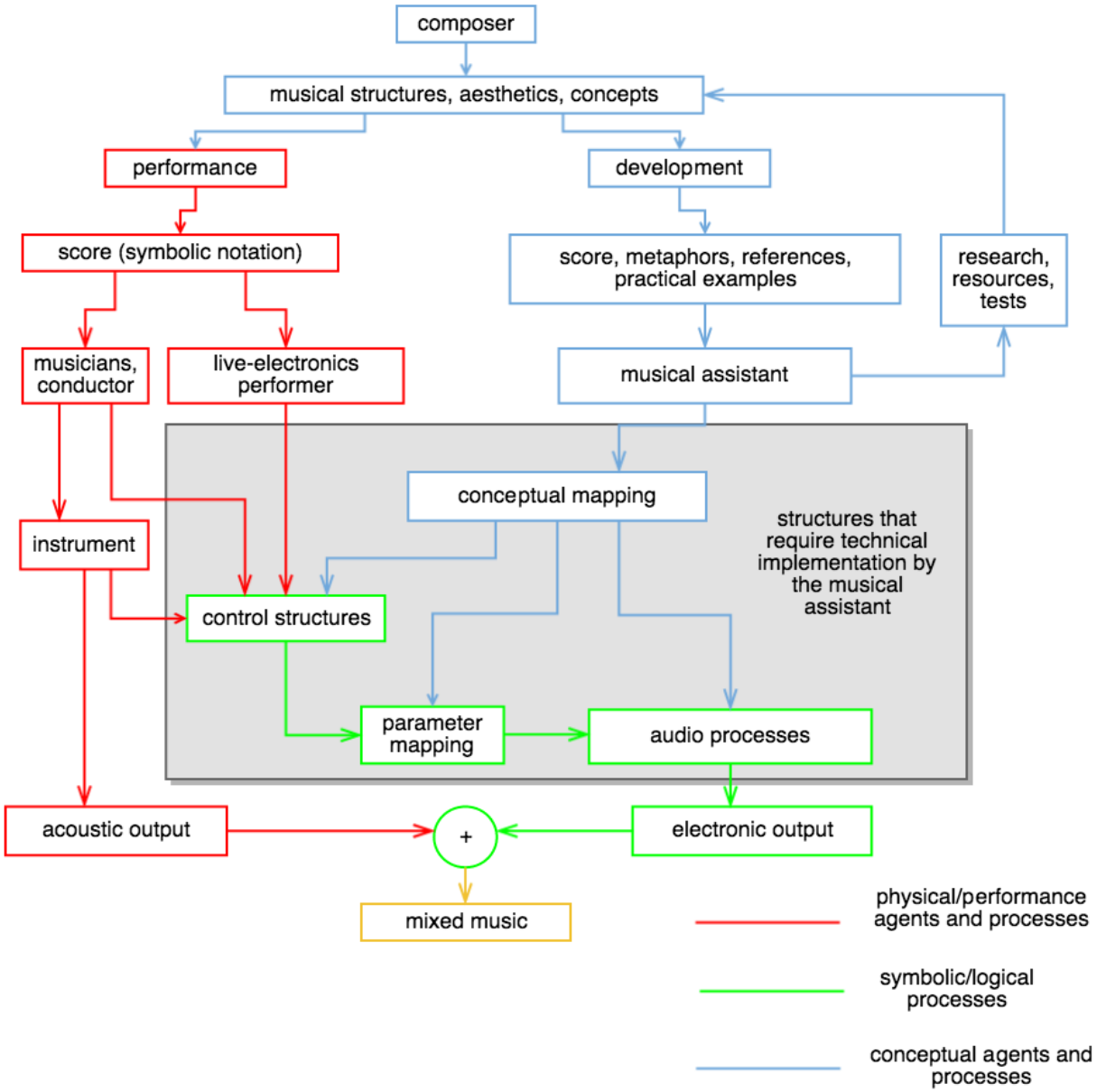


Fig. 2.1: Agents and processes of a collaborative mixed music work.

by the composer.

2.2.3.3 The Musical Assistant

When composing a new piece that involves technology and acoustic instruments, the composer is faced with the challenge of articulating his aesthetic ideas with the required technical implementation. If impregnated by a speculative spirit, with the goal of achieving the

sounds and interactions that are set and conceptualized for that particular music composition, the composer must overcome problems that are much more related to computer and engineering than to music composition. Dealing with such complex technical tasks demands a significant amount of time that is seldom available, and more importantly, that requires skills and knowledge which are rarely part of a composers background. Consequently, the sophistication and effectiveness of the final results of such an imbrication between musical structures, instrumental and electronic sounds is directly affected by the composers proficiency in dealing with technology.

One strategy that is commonly adopted by composers employs the use of existing user-friendly and well established tools to create a sort of patchwork of audio processes that best fulfill the desired sound aesthetic. Using pre-existent tools usually restricts the exploration to the manipulation of the parameters that where predetermined by the developers, which in turn, are usually focused on a broad and generic usage driven by commercial purposes. This type of strategy privileges an heuristic (i.e experimental) approach of trial and errors in which the results seldom reflect the initial objectives imagined by the composer (Risset, 1999). Conversely, a deterministic strategy implies that in order to achieve the desired artistic and aesthetic results, the composer must be able to shape or modify existing tools or design and develop new ones dedicated to a specific creative idea.

These two approaches are deeply related to the concept of *bricoleur* and *engineer* as described by Claude Lévi-Strauss in his work *The Savage Mind* (Lévi-Strauss, 1966). The *bricoleur* develops his work by taking advantage of a collection of tools that is already established and uses these tools for any given job. The collection of tools used by the *bricoleur* is not defined by a specific project or task, they are generic and even though this represents a limitation, different problems can be solved by using the tools in a different way or with a different perspective. The *engineer*, on the other hand, extends the collection of tools by creating new ones from "raw" material, by means of a deterministic approach and deep understanding of the concepts that lie underneath a given problem.

Lévi-Strauss concludes that it is not possible to presume superiority of one over the other. Applied to our case, this conclusion highlights the fact that the choice of an heuristic

or deterministic strategy does not dictate the final outcome of the musical piece. However, as stated by Boulez, a completely heuristic strategy in the search for musical solutions leads to a superficial and immediatist reflexion about the relationship between the sound material and musical ideas, and in his view, this goes against an organic and inventive artistic progress. In his words:

> If invention is uninterested in the essential function of the musical material, if it restricts itself to criteria of temporary interest, of fortuitous and fleeting coincidences, it cannot exist or progress organically; it utilizes immediate discoveries, uses them up, in the literal sense of the term, exhausting them without really having explored or exploited them. Invention thereby condemns itself to die like the seasons (Boulez, 1978).

With that in consideration, a very common strategy historically adopted by composers is to develop the composition in collaboration with a technology expert, which in this context is commonly credited as musical assistant (MA) (Poletti et al., 2002). Some of the most important live electronic music works have been developed by this kind of collaboration:

- *Répons* (1981) by Pierre Boulez, for six instrumental soloists, chamber orchestra and live-electronics, commissioned by the Southwest German Radio for the Donaueschingen Festival and premiered there on October 18, 1981, by the Ensemble InterContemporain conducted by Pierre Boulez and technical support (musical assistants) provided by IRCAM, is considered one of the most important and iconic works developed at IRCAM due its musical and technological innovations (Gerzso, 1984).

- *Jupiter* (1987) by Philippe Manoury, for flute and live-electronics, premiered on April 25, 1987 at IRCAM, technology developed by Miller Puckette, Cort Lippe and Marc Battier, is considered a ground breaking work for being the first presenting real-time pitch tracking and score following (May, 1999).

- Ofanìm (1988) by Luciano Berio, for female voice, 2 children's choruses, 2 ensembles and live-electronics, technology developed at Tempo Reale Florence, premiered on

June 25, 1988, at Museo d'arte contemporanea Prato, is one of the most symbolic mixed music works of the composer, due to his exploration of the relationship between the scenic performance and audio spatialization.

In the development of an artistic work that is carried on by an interdisciplinary collaboration between composer and musical assistant, an exchange of information, ideas and concepts takes place and becomes the interface through which the artistic work will be "materialized". In this scenario, the role of the musical assistant is to enlighten the composer on the current state-of-art in computer music technologies, understand the artistic and aesthetic necessities of the composition and carry on the process of research and technical implementation that will be required.

Even though arts and computer science have been in contact and have mutually influenced each other since the advent of multimedia technology in the 1960s, there is still very little research around the topic of software engineering for arts (Trifonova et al., 2009). Particularly in the case of the musical assistant, despite the fact that the topics related to musical composition, aesthetics, musical informatics and audio processing are very well-established in the academic and commercial worlds, the work and practice of a musical assistant is not yet formally developed in terms of formalized theory and practice methodologies (Zattra, 2006, 2013).

2.3 Live-electronics System

On a technical perspective, the electronic part of a mixed music piece is everything that is not a pure acoustic sound generated by an acoustic instrument performed by a musician on stage. And consequently, the *live-electronics system*, or simply the *live-electronics*, is the technical implementation in the form of computer software and dedicated electronic hardware (when necessary) of the live electronic part.

2.3.1 Definition

Defining the electronic part of a musical piece on a pure technical perspective avoids the problems that would arise from a definition conceived from a musicological or compostional perspective. As well stated by Stephen Montague (Montague, 1991), the definition of live electronic music on a composers point of view is very intimate. Different composers have different insights about the underlining concepts and implicated methodologies that come with the term.

The context of our definition is equally relevant. By contextualizing the definition presented here in the erudite (or serious) contemporary music domain, whose creative and aesthetic musical ideals rely on formal composition and writing (Nicolas, 2002), more precisely on the electroacoustic music genre, we observe a fundamental characteristic: In this context, the electronic and technological means are an intrinsic part of the compositional process.

A good example of this characteristic can be observed in the use of synthesizers (of any kind) in the musical practice. Synthesized music as it first appeared in the electronic music context in the early 50's presented a very deep relation between the musical concepts and the technological means used to implement that music. Synthesizing sounds was not just a creative use of available technological resources but it was also a very strong conceptual statement regarding the new paradigms and aesthetics of music composition. As synthesizers became less expensive and easy-to-use, they started to permeate other musical genres, being used with multiple purposes that had nothing or very little to do with the initial ones. The use of synthesis in Stockhausen's early compositions or Risset's compositions using paradoxical sounds (Risset, 1989), and the use of synthesis for simulating string instruments in a pop music hit, present a common technical tool, but very distinct objectives and context.

Furthermore, from this example we can observe two very distinct approaches to the use of music technology: One uses technology at the structural and conceptual levels, and the resulting musical piece is codependent on both the artistic and scientific aspects of the composition process. The other displays a more casual or pragmatic use of available techno-

logical tools (Rowe, 1999). Both approaches present merits and are pertinent to the context that they are inserted. There is no superiority of one over the other, and each approach has a different influence on the music society. The first is responsible for generating low-level structural and conceptual knowledge and technologies. The latter is responsible for disseminating the concepts and technologies over a wider range of users (consumers) and hence establishing (or not) a technology and its associated tools and paradigms.

In conclusion, in the context of this Thesis, a live-electronics system refers to the technical implementation of the electronic part of a mixed music composition, where the music (both instrumental and electronic) undertake a composition process prior to the performance (as discussed in section 2.2.1). The music composition and the live-electronics system are fixed and the performance is subject to interpretation.

2.3.2 Live-electronics System Typical Structure

The live-electronics system of a mixed music composition functions as the counterpart to the instrumental score, therefore it must present an analogous structure to manage musical events and their respective dynamics and progression over time.

A musical event represents a group of instructions (one or more simultaneous instructions) that must be put into action at a determined instant of time and have a global or localized affect in the performance. An event can hold any kind of instruction relevant to the performance, ranging from purely musical information such as pitch, duration, intensity, timbre, accent, tempo, dynamics, embellishments, among others, to purely logical such as repetitions, change of cleff, and so forth. By this definition it is possible to describe a music composition by a succession of instructions organized in events (Buxton et al., 1978).

In the case of instrumental music, an event is a musical structure such as phrase, period, or similar and the instructions, represented by symbols on the score, are the necessary information for the musician to perform that musical event. The instructions represent the information that is required for the musician to perform the respective event. In figure 2.2 a piece of instrumental score written in common practice notation is displayed. The score demonstrates the occurrence of a musical event, in this case a period[1], and a succession of

instructions numbered 1 to 8.

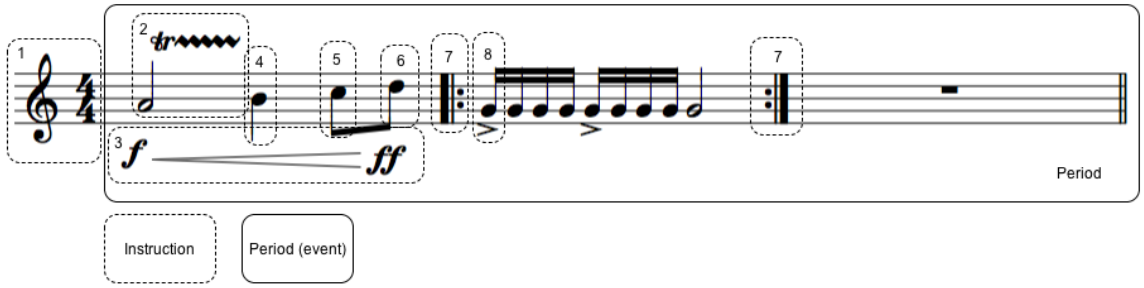


Fig. 2.2: A traditional instrumental score displaying a musical event (period) and a succession of instructions.

Here it is possible to observe distinct types of instructions such as logical (1, 7) and purely musical (2, 3, 4, 5, 6, 8).

- Instruction 1:
  contains logical information (clef and time signature) used to designate how the musician should read the notes in that music excerpt. Thus in this case this instructions have a global effect.

- Instruction 2:
  contains musical information (pitch, duration, embellishment); it tells the musician to play the A note with a duration of half-note embellished by a thrill.

- Instruction 3:
  contains musical information (dynamics); it displays a dynamic progression (forte to fortissimo).

- Instructions 4, 5, 6:
  contains musical information (pitch duration).

- Instruction 7:
  contains logical information; it tells the musician to repeat that specific bar.

[1] A period is a short musical structure that represents a complete idea (such as a melody), usually part of a bigger structure formed by several periods.

- Instruction 8:
  contains musical information; it tell the musician to play the G note with an (embellishment) accent.

The logical and musical instructions are sufficient for the musician due to the fact the musician has a previous knowledge acquired by years of training and performance experience that enables him to decode the symbols and context into an expressive interpretation of the composition. Furthermore, the timbre is already preset by the instrument itself, taking out a huge part of the information that would otherwise needed to be precisely symbolized in the score.

The live-electronics system must perform an analogous task as that of the score-musician-instrument. It must contain all the information needed to perform musical events at precise instants and with precise control over musical and logical parameters. For that reason, a typical live-electronics system is organized into two main structures: audio processes (defined in section 2.3.5) and control structures, which relate to the sonic and structural level of the composition, respectively. The coupling between this two main structures is realized in an intermediary parameter mapping layer (figure 2.1).

Figure 2.3 displays the organization of the control structures, going from the highest level (Global control structure) to the lowest level (parameter values that obey a function of time, depicted inside the local control structure as a multicolored graph). These structures are explained in detailed in the following sections (sections 2.3.3, 2.3.4).

This separation of control and audio process has been used since Matthew's MUSIC III software of the MUSIC N family (as discussed in section 2.1). Then, it was based on the concept of combining the virtual electronic instruments into "orchestras" and managing their parameter values over time using a "score" file. The implementation of this paradigm had a major influence in the compositional aspect of computer music, by creating a system that mirrored the instrumental composition process. And also on the technical aspect of audio programming, by implementing the concept of audio processing modules (named unity generators (ugens)) that is still used by current state of art audio programming languages (Chadabe, 1997; Honing, 1993).

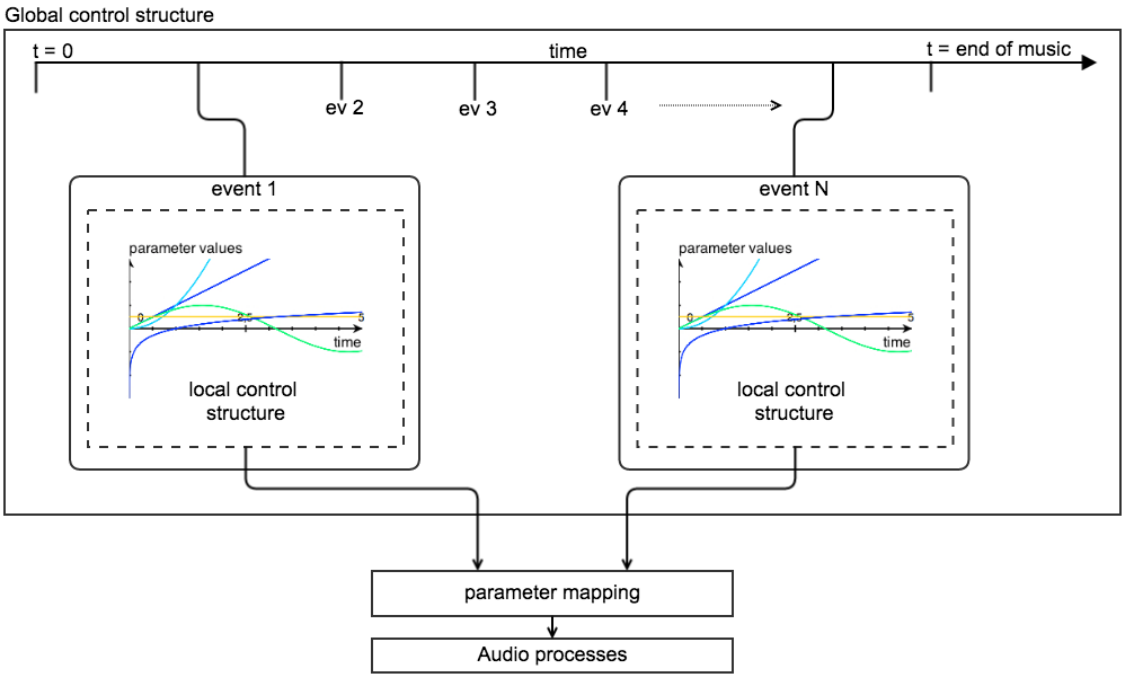


Fig. 2.3: Control structures of a typical live-electronics system.

Control structures are the mean for controlling the behavior of the electronic sounds during the performance. Each audio process implemented in the live-electronics system requires a specific set of parameters to control its behavior (timbre and dynamics). Thus, each audio process requires a control structure, here denoted as local control structure. Additionally, the live-electronics system requires an overall control structure that is responsible for triggering and organizing the different individual processes that take place (events), here denoted as global control structure. Essentially, global control structures are responsible for synchronizing the instrumental part with the electronic part of the music, while local control structures are responsible for controlling musical parameters such as timbre and dynamics.

A typical live-electronics system is organized in events (or cues) that need to be managed during the performance. It is common practice to indicate the live-electronics system events in the instrumental score. Figure 2.4 displays an excerpt of the score from *Quaderno* (2005) by Flo Menezes[2]. Each event (marked with a yellow painted rectangle and the name of the audio process that take place) contains the information required to activate (or deactivate) and control the audio processes that need to perform at that instant. Events can be as simple

as playing a sound file (most common type of event in mixed music for instrument and "tape"). And as complex as activating and assigning parameters for several parallel audio processes and their respective control variables.

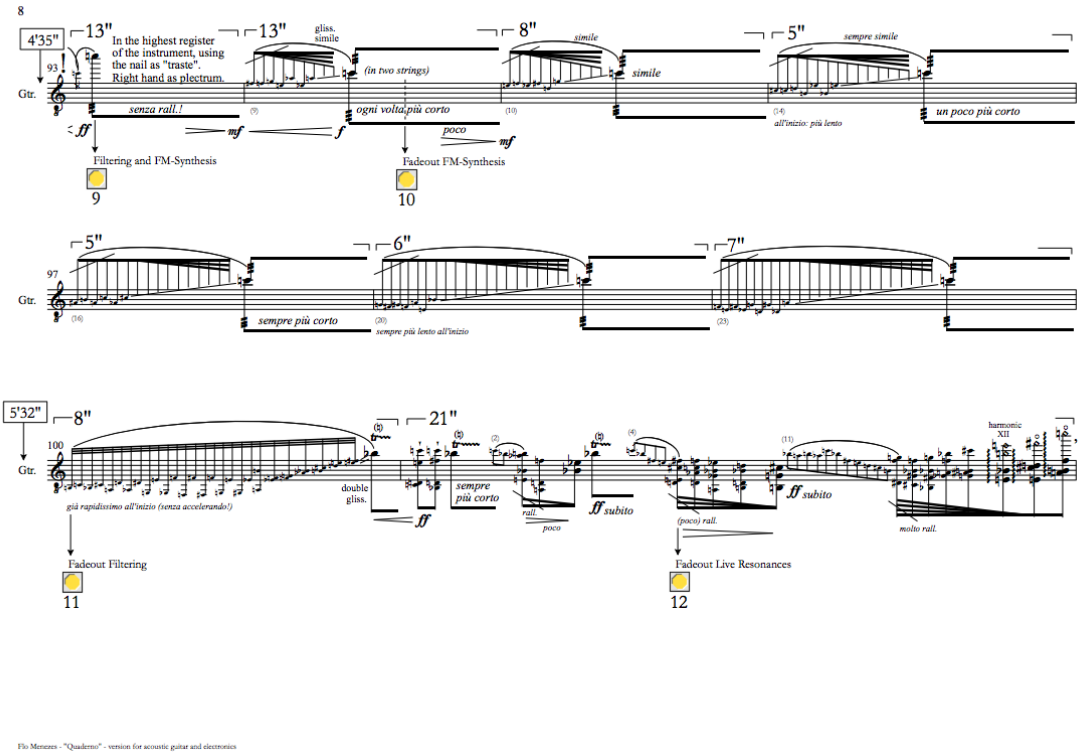


Fig. 2.4: Excerpt from *Quaderno* (2005) by Flo Menezes. Displaying live-electronics events 9 to 12

It is important to notice that in the instrumental part, multiple events can occur simultaneously (polyphonic music) and independently. In the live-electronics system, due to the fact that an event represent at the same time a musical structure and a logical cue used for synchronization purposes, only one event is triggered at a time. Nevertheless, an event can superpose the previous one if the duration of the first is longer than the time lapse between triggers.

[2]see apendix A, composition: 1

2.3.3 Global Control Structures

There are three main global control structures strategies: manual triggering, score following and timeline-oriented control. Each strategy presents advantages and disadvantages. The choice of strategy should be done to meet the necessities of the overall music performance, taking in consideration all related parties (composer, musician, conductor, digital performer, technical staff).

Additionally, there is no mutually exclusive rule, thus it is possible to combine different strategies and even alternate between them during the performance.

2.3.3.1 Manual Triggering

Manual triggering is the simplest and most common strategy for global control. In this strategy events are manually triggered by a digital performer who follows the instrumental score marked with events (see figure 2.4) and triggers each one at their precise instant. Typically this is performed by the composer and/or the musical assistant, and the trigger is done directly (mouse/keyboard) in the computer running the live-electronics system software. Other common approach is to assign this job to the musician (or conductor). Musicians can assert command over the global control using physical controllers such as a foot-switch or similar mechanical interface.

Manual triggering provides a precise global synchronization between the electronic and instrumental parts, and it allows easy correction of event placement in the case the instrumental part is mistaken or an event is erroneously triggered.

In cases that demand a highly accurate synchronization (such as a precise simultaneous note attack), it is best to delegate the trigger action of that specific event to the musicians or conductor.

Mixed music works that involve a large number of acoustic instruments such as orchestras and ensembles tend to use this approach due to the fact that it is the most simple to perform and presents a good balance between reliability and accuracy.

2.3.3.2 Score Following

A score following systems performs a real-time analysis of the incoming audio input from the acoustic instruments and tries to identify the instantaneous position of the performance by comparing the analyzed data to the music score. Thus providing a continuous synchronization of instrumental and electronic parts and an automatic triggering of events.

Score following is a highly complex computational task and it is currently a widely researched technique that is closely related to the domain of Music Information Retrieval (MIR). Nevertheless, current score following systems are still very limited and problematic, specially in the cases where the instrumental input consists of very complex and polyphonic spectrum.

The advantage of using a score following system is that (in theory) it permits a complete synchronization of the instruments and live-electronics giving the performer total freedom (from a technical perspective) in what regards the tempo and timing of the music. On the down side, it presents many technical limitations and therefore conditions the instrumental writing. Also, it is very susceptible to the performers mistake and unpredictability, and therefore it usually requires human supervision (Orio et al., 2003; Puckette and Lippe, 1992).

Score following succeeds in works for solo instrument and live-electronics in cases where the acoustic instrument presents a very stable timbre over its spectral and dynamical range. Specially, in the case where the instrumental input can be effectively converted into MIDI (Rothstein, 1992). This particular scenario is the case of important compositions using this technique such as *Jupiter* (1987), for flute and live-electronics and *Pluton*, for piano and live-electronics by Phillipe Manoury and *Duo for One Pianist* (1989) for the Yamaha Disklavier MIDI piano, by Jean-Claude Risset.

2.3.3.3 Timeline-oriented Control

A timeline-oriented control refers to the process of using a fixed tempo structure such as a timeline or sequencer where events are fixed in a predetermined position in time. The timeline is performed (played) in a linear progression of time, and events are triggered in

their respective time instants (Zadel and Scavone, 2006). This approach is very similar to the mixed music for instruments and fixed medium, where an “event track” substitutes the audio track.

The advantages of this approach is that it doesn’t require supervision or manipulation during the performance. The disadvantage is that it presents the same problems of the fixed medium mixed music compositions, where each event has a predetermined position and duration that performs independently from the instrumental performers.

2.3.4 Local Control Structures

In order to discuss the local control structures, an example from the composition *Madrigal*[3](2014) by Marcus Siqueira, for 11 strings guitar and live-electronics, is presented. In this work, the live-electronics system events are triggered manually by the player using a foot-switch. Each event triggers a set of instructions for the live-electronics system software. Figure 2.5 displays events 5 and 6 of the live-electronics system.

Event number 5 triggers the following instructions: Activate cumulative freezing effect with 4 independent layers, fade in cumulative freezing effect in short period of time, start spatial trajectory of each freezing effect layer. Event number 6 triggers the following instructions: Fade out all layers of freezing effect in approx 7".

By analyzing the instructions that are triggered in event 5, it is possible to see that it involves three distinct audio processes[5]: a freezing effect (which in this case is implemented using a phase-vocoder), the gain (volume) control of the freezing effect and a sound spatialization algorithm. Moreover, it involves four independent instances of each of these processes. When the event is triggered each of these audio processes must perform a determined trajectory from a initial (current) state to the desired state. In this particular example, the most simple itinerary is performed by the gain control of the freezing effect. The gain control must go from 0 (zero = mute) to a pre-determined value x > 0 (audible) in a determined time frame. Likewise, the sound spatialization variables must be controlled so that

[3] see apendix A, composition: 2

[4] Some of the instructions are explained in the score and some are embedded in the live-electronics system software.

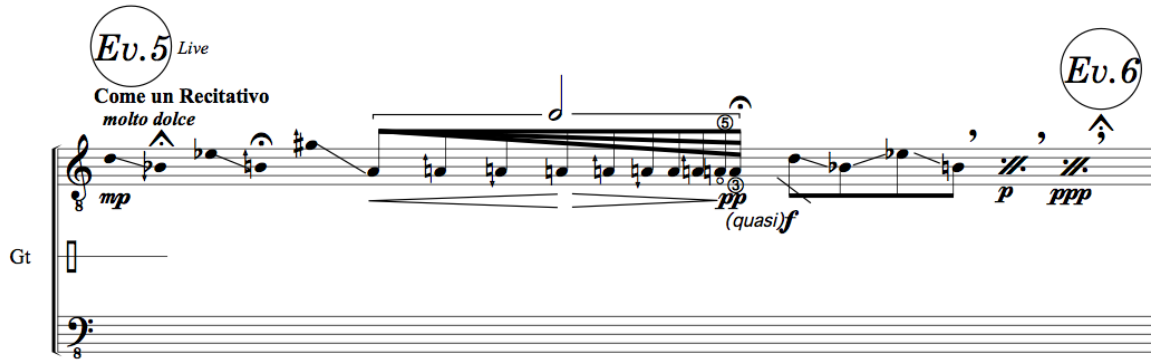


Fig. 2.5: Displaying live-electronics events 5 to 6. Event 5 instructions[4]: Activate cumulative freezing effect with 4 independent layers, fade in cumulative freezing effect in short period of time, start spatial trajectory of each freezing effect layer. Event 6 instructions: Fade out all layers of freezing effect in approx 7".

the sound performs a spatial trajectory and, the freezing effect must have its parameters (pitch, modulation, among others) controlled so that the produced timbre respects the desired sonority aesthetics. The control over all these parameters is managed by local control structures.

In non-real-time electroacoustic music (such as Tape Music), this control is done in deferred time, and composers are free to make as many iterations as necessary to achieve a final sonority. In the case of mixed music that uses real-time operations to react and interact with the instrumental input, the control over the parameters must be executed in real-time during the performance.

To achieve complex electronic sounds that can match the expressiveness, articulation and spectral complexity of instrumental sounds, a sophisticated network of audio processes is required. Consequently a significant number of parameters need to be controlled in real-time. In our context, where music composition precedes the performance, the problem of controlling parameters so that the composed (desired) sound aesthetics is performed with the highest possible fidelity and at the same time aggregates the expressiveness and interpretation of the musicians, is of fundamental importance (Menezes, 2002; Risset, 1999; Stroppa, 1999).

Marco Stroppa describes this problem by stating that a composition that involves live-

[5]see definition of audio process in section 2.3.5.

electronics has two distinct states: a “composed state” and an “interpreted state”. The distance between both is described as the “interpretative potential” and it is determined by the strategy for controlling the audio process parameters. Hence, parameters that are set in the composed state and are reproduced without any degree of influence during the performance, present no interpretative potential; parameters that are influenced or directly react to the instrumental input (or some other form of interaction) present a high interpretative potential (Stroppa, 1999).

A good balance between fidelity to the composed aesthetics and interpretative potential is achieved by combining absolute and relative values for the variables that govern the sonority produced by audio processes (Manoury, 1998). Absolute and relative values relate to automated and non-automated local control structures respectively.

2.3.4.1 Automated Control

An automated control strategy implies that the values of an audio process parameter control variable are predetermined and executed without any direct human manipulation. Values are assigned as a function of time and require only an initial trigger for generating its output, there is no direct manipulation of the output values.

Automated control is very useful for assigning values for parameters that require a high level of precision (e.g. a precisely timed crossfade between several distinct audio processes); performing trajectories that would otherwise be very complicated to replicate manually (e.g. complex 3D sound spatialization trajectories); performing a complex trajectory in a very short period of time (e.g. controlling the dynamic envelope of individual components of an additive synthesis in order to generate a bell-like sound); assigning values using stochastic processes.

Applying an automated control does not mean that the values assigned to the respective parameters are fixed in time or that they do not present any expressiveness. An automated control can be implemented with complex algorithms, mathematical functions and manually designed trajectories. The terms automated and absolute imply only that in any performance of the respective music work, the parameters controlled with this kind of strategy are going

to perform with the same behavior, independent to the instrumental performance.

2.3.4.2 Non-automated Control

A non-automated control strategy (or manual control strategy) implies that the values of an audio process parameter variable is controlled by a performer agent (real or virtual).

There are two basic non-automated control implementation strategies: using relevant information extracted from the instrumental audio input (audio feature extraction); using control interfaces that can be virtual such as software's graphical user interface, or real such as digital instruments, midi controllers and generic sensors passing through an analog to digital converter (ADC).

Audio feature extraction

Audio feature extraction refers to digital signal processing techniques for extracting relevant information such as pitch, spectral components, noiseness, amplitude, among others, from audio signals. The basic idea in this technique is to use meaningful perceptual information retrieved from the instrumental audio input and use this information as an input to control an audio process parameter variable.

Audio feature extraction is currently one of the most researched subjects in computer music, and many works related to the specific topic of using audio features to control audio processes in real-time electroacoustic music have been recently published (Bullock, 2008; Hoffman and Cook, 2007; Jehan and Schoner, 2002).

This strategy allows for effective coupling between the instrumental expressiveness and the electronic sounds, specially in the cases where the intention of total synchronization between both sources is present. In that matter, techniques such as pitch tracking and envelope following (amplitude tracking) are largely used and most audio programming environments offer out-of-the-box implementation tools. Nevertheless, the number of features that can be tracked with enough precision is usually significantly smaller than the number of variables that require control (Lee and Wessel, 1992). And therefore a sophisticated parameter mapping strategy is required.

In the specific case of mixed music, using audio features such as pitch and amplitude to control electronic sounds that are of complete different spectral quality, is rather complicated due to the fact that the acoustic source can usually be heard and therefore the resulting sound is a coupled superposition of instrumental and electronic, which in turn is very difficult to work with (Stroppa, 1999).

Physical interfaces

Using physical interfaces such as the mouse, knobs, sliders and pedals is a routine practice for those who work in a music studio. There is a great number of affordable devices that can easily connect to the live-electronics system software via MIDI or OSC and function as a generic controller that can be attached (mapped) to any number of audio process parameters. However, this type of standard interfaces have rather limited expressive potential, and controlling several parameters at the same time presents a considerable challenge in a real concert situation.

One strategy to overcome this issue is to delegate the control to the musicians by augmenting their instruments with sensors or employing completely digital ones (Tindale et al., 2005). The first case refers to the use of sensors such as pressure, breath, gyroscope, bod tracking, among others, to extract physical gestures from the musician and translate it into the electronic sounds. The second case refers to the use of completely digital instruments that do not have acoustic components, but can send musical information such as intensity and pitch to be converted into electronic sounds.

In any case, employing physical interfaces present the same problems as the audio features extraction, and aggregate pragmatical problems such as the need for dedicated performers in the concerts; calibrating sensors takes a lot of time in the rehearsal and concert preparation (time is usually very scarce); using digital instruments require that the musicians have enough practice time with the instrument and the electronic sounds (the live-electronics system).

2.3.5 Building Blocks

As opposite to the early years of electroacoustic music, composers are now able to combine any kind of technological technique and aesthetic in the same musical piece. There are no technical nor conceptual restrictions. If in the early days there was a rivalry between those that used synthetic sounds and those that used concrete sounds (i.e recorded and sampled sounds), today, combining elements from different approaches and aesthetics is no longer frowned on. In addition, the advances regarding human-computer interaction and the development of digital instruments and controllers offers a vast variety of real-time interaction possibilities for live performances.

A typical live-electronics system of an electroacoustic mixed music piece is based on the combination of distinct compositional, technical and technological methods and the building blocks that constitutes the live-electronics system can be divided into three main categories of the digital signal processing domain: synthesis, sampling and digital sound processing (or digital audio effects) (Manoury, 1998).

2.3.5.1 Sampling

Sampling consists of recording a sound and storing it in the computer memory in audio format. The sampled sound can then be reproduced and/or manipulated (Puckette, 2006). This is the digital equivalent of the most common technique of the concrete music, which used recording and editing as the central tool for creating and diffusing electronic sounds.

Despite the fact that a sampled material is rather inflexible and the possibility of control without the aid of a complex chain of audio processes is limited (playback speed and direction), sampling is still widely used in current live-electronics system due to the fact that it allows the use of sound material that would be very difficult to be achieved by synthesis (Manoury, 1998).

One of the most common applications of sampling in a mixed music is, during the performance, to record a fragment of the instrumental part and using it later (with or without transformations) in a distinct instant of the music piece. Evidently, the recording could be done in deferred time, in studio, however, the possibility of recording it during the perfor-

mance itself, provides a more contextualized sound, that reflects the instantaneous expression and timbre of the musician and instrument in the concert room.

2.3.5.2 Synthesis

Digital sound synthesis consists of modelling the pressure function of a sound wave using mathematical equations. Synthesis can be used to simulate real instrumental and natural sounds and also to create and design completely unrealistic ones (Roads and Strawn, 1985). The most important feature of digital sound synthesis if the fact that it offers the possibility of controlling all aspects of the generated sound, by controlling the spectral content (frequencies) and dynamic content (dynamic envelope).

One of the most used synthesis techniques is "additive synthesis". It is based on the principle that any timbre can be achieved by superposing individual spectral components, usually generated by sine wave oscillators. This technique has been used since the beginning of computer music to model instrumental sounds, understand instrumental timbre spectrum and to achieve novel timbres and sounds. In theory, additive synthesis is capable of generating any kind of sound, by meticulously selecting the spectral components and the respective dynamic envelope of each component. However, in practice, this represents an enormous amount of data manipulation that consumes a lot of time and computer processing (Roads and Strawn, 1985).

Frequency modulation, a digital synthesis technique developed by John Chowning (Chowning, 1977) based on the equation and principle of carrier and modulator as used in radio transmission, revolutionized computer music due to the fact that it creates complex spectra in a simple and elegant way. This is crucial in terms of computer processing optimization and also it allows for more intuitive and perceptually sensible control.

Other important and well established synthesis techniques are: waveshaping synthesis, where a sinusoidal input oscillator is distorted by a nonlinear processing function (waveshaper) and thus the output presents a more complex spectra than the input signal (Roads, 1979); subtractive synthesis, where the desired timbre is achieved by filtering out spectral components of a complex spectra input (Puckette, 2006); physical modelling, a technique

used to reproduce real acoustic sounds by simulating the mechanical principles that generate the sound (Smith, 1992).

2.3.5.3 Digital Audio Processing

Digital audio processing (also known as digital audio effects) represents the category of signal processing algorithms that alter specific characteristics of the input signal, which in turn can consist of the instrumental/vocal acoustic input and the audio generated by synthesis or sampling.

The most common processing techniques are: filtering, distortion, modulation, spatialization, delay-based effects such as flanger and echo, compression, pitch shifting, time-stretching, among others (Viers, 2011).

Digital audio processing is a fundamental part of every live-electronics system, it represent the tools for shaping the "raw" spectral material into the desired aesthetic outcome.

2.3.5.4 The Audio Process

Advances in computer software technology and performance, fomented the development of many hybrid techniques that merge sampling, synthesis and processing with highly complex real-time control possibilities. Some examples of these hybrid techniques are: analysis/resynthesis, phase-vocoding, granular synthesis, among others.

For the purpose of generalization, we will use the term *audio process* when referring to any of the digital signal processes presented in the previous sections (2.3.5.1, 2.3.5.2, 2.3.5.3), assuming that it can be used as module (or component) in a complex signal processing network. Ergo a module that is implemented using several distinct audio processes is also an audio process (e.g.: reverb effect, typically implemented with delay lines and filters, will be referenced as an audio process. As well as a delay and a filter are also referenced as audio processes). Essentially, all synthesis, sampling and processing techniques and algorithms will be referred as *Audio Process*.

2.3.6 Mapping Structures

The term mapping usually refers to the act of creating a systematic correspondence between elements of two distinct groups (source and target). The term is used in several distinct contexts and thus the elements and groups that represent the source and target vary significantly.

If looked in the mathematical sense, the term mapping usually refers to a mathematical function, which in turn is the relation between a set of input elements to permissible output and each input relates to a unique output (Piskunov and Yankovsky, 1974). Figure 2.6 displays a mathematical function: $f : X \rightarrow Y$, where each element of X relates exclusively to one element of Y.

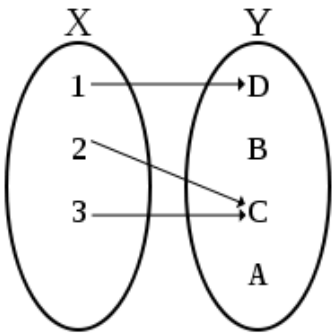


Fig. 2.6: Mathematical function:
$f : X \rightarrow Y$

In the scope of computer music the term mapping is used to refer to the correspondence between control inputs (source) and audio process parameter variables (target) (Hunt and Wanderley, 2002). In this context we will refer to mapping as *parameter mapping*.

Equally significant is the use of the term mapping to represent a conceptual or cognitive metaphor. Here the term mapping represents the relationship between two distinct concepts in the metaphorical process (Kovecses, 2002). We will refer to this as *conceptual mapping*.

2.3.6.1 Parameter Mapping

Parameter mapping is the act of coupling the values of a controller input to the parameter values of an audio process. In a live-electronics system, the parameter mapping layer connects the control structures to the audio processes (as depict in figure 2.1).

The parameter mapping serves multiple purposes. It must deal with simple and straight forward problems such as converting range of values (e.g. converting a MIDI continuous controller value with value range: 0 - 127, to linear amplitude values ranged: 0.0 - 1.0), and at the same time it must deal with complex problems such as connecting multidimensional raw mathematical data to acoustically perceptual parameters (e.g. a reverb process implemented using a network of delay lines with feedback in parallel and in series can have its perceptual parameter *reverberation size* modified by adjusting the values of the delay time, feedback amount and buffer size of each delay component of the network. Thus, if only one variable is assigned to the parameter *reverb size*, this variable must be mapped to the variables that are effectively responsible for creating the perceptual change).

Furthermore, parameter mapping presents complex challenges when there is a direct intention of converting musical and physical gestures from a performer into electronically generated sounds. Joel Chadabe discusses this problem by comparing the strong relationship of a musician and an acoustic instrument to the relationship between a physical controller and an electronically generated sound. In the former, the musician has very deep and precise control over the generated sound by acting in the physical element that generates the sound (e.g. the bow and strings of a violin). In the later, there is no physical or acoustic relationship between the input and the generated sound. Also, in the case of electronically generated sounds, the link between the controller and the generated sounds can be active, and the input information from the controller can be drastically modified in the computer, breaking the causal link between the human input and the output sound (Chadabe, 2002).

Marco Stroppa goes even further, in his words:

> The interaction between a performer and his or her instrument is so tightly coupled to the physical behavior of the instrument itself, that I do not see why, for instance, extracting some phrasing characteristics from a clarinet and applying them to a cello should sound anything but clumsy. It is even harder with synthesized sounds, since then the acoustical instrument, being usually superposed to the processed sounds, cruelly pinpoints the difference, or one would have to store the analysis parameters and use them later. But then the relationship with

the originating gesture will probably be unhearable. (Stroppa, 1999)

The problems related to parameter mapping have been widely studied in the computer music domain and several mapping strategies have been proposed.

Mapping strategies can be segmented into three main categories (Hunt and Wanderley, 2002)

- One-to-one:
  One control value relates to one audio process parameter value (e.g. one fader controls the master volume of the live-electronics system).

- One-to-many
  One control value relates to several audio process parameter values (the case of the previously presented reverberation time example).

- Many-to-one
  Several distinct control values are combined to control one audio process parameter value (e.g. the pitch of an oscillator is controlled by the combination of the tracked pitch of several acoustic instruments).

Additionally, there are distinct perspectives on the way a mapping strategy should be organized and implemented. Chadabe (Chadabe, 2002) proposes an *hierarchical* and *network* organization schemes for combine several distinct input and output variables. Nort (Nort et al., 2014) presents a *functional* perspective, which focus on the mathematical operations used in the correlation between variables.

2.3.6.2 Conceptual Mapping

Composing the players and the instruments is an analogy for implementing audio applications using digital signal processing building blocks and control structures. This implementation process represents at the same time technical and artistic matters. It serves as the output of the artistic process but at the same time it must serve as a platform for creative exploration and experimentation.

The implemented electronic players and instruments are, in fact, what we previously described as the live-electronics system, and the connection between the artistic concepts and audio building blocks that are ingrained in its core, is what we define as the *conceptual mapping*.

The decision of which type of audio processes (in section 2.3.5.4 a definition of audio process is presented) to use and how to control their parameters must be a direct consequence of the musical ideas and concepts elaborated by the composer. The factors that influence these decisions are the conceptual relationships between the acoustic and electronic sources and the sonority aesthetics of the final output.

In the scope of electroacoustic mixed music works, Mike Frengel (Frengel, 2010) proposed a multidimensional framework to analyze the possible relationships between the acoustic and electronic sources from a musical perception point of view. In his framework he organized those relational aspects into nine primary classifications: segregational, proportional, temporal, timbral, behavioural, functional, spatial, discursive and pragmatic. In what regards the technical implementation, the most important relationships are the timbral and temporal ones, representing the spectral and time domains, respectively.

The spectral domain relations can be of:

- Conservation of the acoustic spectrum. The electronic part takes advantage of the acoustic spectrum and the audio processes infer mostly on the time domain.

- Transformation of the acoustic spectrum through the use of effects such as distortion, flanging, filtering, etc.

- Expansion of the spectrum by derivation of the original acoustic spectrum through the use of analysis and resynthesis techniques.

- Addition to the acoustic spectrum by the use of synthesis techniques.

The time domain relations can be characterized as:

- Synchronous. The electronic part reacts instantaneously to the instrument input with the minimum possible latency. (Direct effects such as distortion, flanger, filters, etc.).

- Isochronous. Characterized by timely reactions to the input (Delays, time-stretching, retrograde, etc.).

- Independent. The electronic part has an independent time occurrence.

The final desired sound aesthetics also present a major influence on the decision of what kind of audio processes will be used. For example, if we consider a combination of spectrum conservation with isochronous time, a possible solution would be to apply a time-stretching effect, which is possible to be implemented using distinct algorithms. A phase-vocoder time-stretching implementation and a granular time-stretching implementation can fulfill the same conceptual choice but the final sonority is substantially different.

It follows that the universe of possibilities that emerge from the combination of all imaginable relationships between the acoustic and electronic parts is immeasurable. Moreover, it is also possible to interpret the different relational aspects as a continuum relational space, where one aspect can transform into another through any desired path. Rigorously, between instrumental layer and electroacoustic sounds there are infinite levels of interactivity going from maximum *fusion* to maximum *contrast*, as discussed by the composer Flo Menezes (Menezes, 2002).

Given that the exchange of information between composer and musical assistant is carried on during the compositional and creative process, the artistic/technical communication between collaborators is achieved (in both directions) by using multiple mediums such as, metaphors, graphical representations, audible examples, musical references, technology references, software applications, proof of concepts, among other.

Metaphorical descriptions and general conceptualization are a common practice in the music composition process (Leman, 2002). Hence it is only natural that this abstract information play a central role in the development of such interdisciplinary and transcendental matter (musically structured electronic sounds). From that source of abstract material the musical assistant must be able to extract critical behaviors, aesthetics and interaction needs. And from these, implement the live-electronics system.

To better explain this act of translation, a real example of the metaphors and analogies used by composers when communicating with the musical assistant is presented:

The term "resonance" is commonly used by composers to describe the desired sonority and musical intention in a musical fragment or passage. In the development of the live-electronics system for *Crase*[6] (2006) by Flo Menezes, the composer used the term "resonance" to express the intent of holding (or freezing) in time a specific note played by the solo instrument and moving it around in space. Additionally, to express his idea of resonance in this specific musical excerpt he portrayed a poetic image, in his words: "...like an ancient chinese master painting a logogram in the air and leaving fading trails of ink...". In another work also by Flo Menezes, *O Farfalhar das Folhas*[7] (2010), the composer used the same term "resonance" to describe an always present and ever changing sound, of synthesized quality, with harmonic components that represent the harmonic profiles and structures of the composition. In the development of the live-electronics system for *A Laugh to Cry*[8] (2013) by Miguel Azguime, the composer used the term "resonance" to describe the intent of amplifying and elongating in time the actual acoustic resonance of a chord played in a grand-piano.

In each of this cases, the term "resonance" was employed in a different context and thus with a different aesthetic and expressive intention. Nevertheless, in all cases, there is a common behavioral intent of grabbing a frame of a musical instant and spreading its spectral contents in time and space. Indeed, it is this behavior that can (and must) be extracted and implemented in the form of algorithms and audio processes. It is therefore, the link between the musical ideal and technical realization. It is the precise definition of *conceptual mapping*.

Just as there is no mandatory path nor stable compositional theory in electroacoustic music (Zattra, 2006), there is no manual or guide for overseeing the musical assistant in the task of extracting concrete implementable matter from abstract ideas. Each composer and each new composition presents a new message, a new meaning to analogies and metaphors. And each of these can be implemented in many different ways. Thus reasoning different styles, discourses, backgrounds and statements requires from both collaborators much experience

[6] see apendix A, composition: 3
[7] see apendix A, composition: 4
[8] see apendix A, composition: 5

in the artistic and technical domain.

Finally, in the musical assistant perspective, the conceptual mapping is completely code-pendent on the implementation methodology. It is not possible to just make intellectual decisions and choices regarding audio processes. A concrete implementation is imperative. The implementation is the mean to experiment and validate the conceptual decisions. It is the most effective way for communicating back to the composer.

2.3.7 Live-electronics System Implementation

An unique mixed music composition often require a unique custom made live-electronics system to perform its electronic part. Furthermore, different concerts of the same composition may require specific configurations or even modifications of the base (original) system. Despite the inherent uniqueness of each live-electronics system, it is possible to observe and extract general attributes that are common to most typical live-electronics system setups.

In this section the most typical live-electronics system hardware setup and software implementation strategies are presented.

2.3.7.1 Live-electronics System Hardware Setup

A typical live-electronics system hardware setup includes a microphone array for capturing the live input from the acoustic instruments, a speaker array for diffusing the sound on the concert room, a mixing console, an audio interface a computer and eventually digital instruments and controllers. Figure 2.7 depicts a typical setup.

The microphones array connects to the mixing console and is then fed to the computer through the audio interface. The microphones array can be fed directly (channel to channel) or by first mixing down the signal (many channels to mono or stereo).

The computer processes the live-electronics system software and sends the audio output through the audio interface into the mixing console. The mixing console distributes the audio to the speaker array.

Digital instruments and controllers connect directly to the computer usually communicating MIDI protocol. Off-the-shelf devices usually present an embedded analog-to-digital

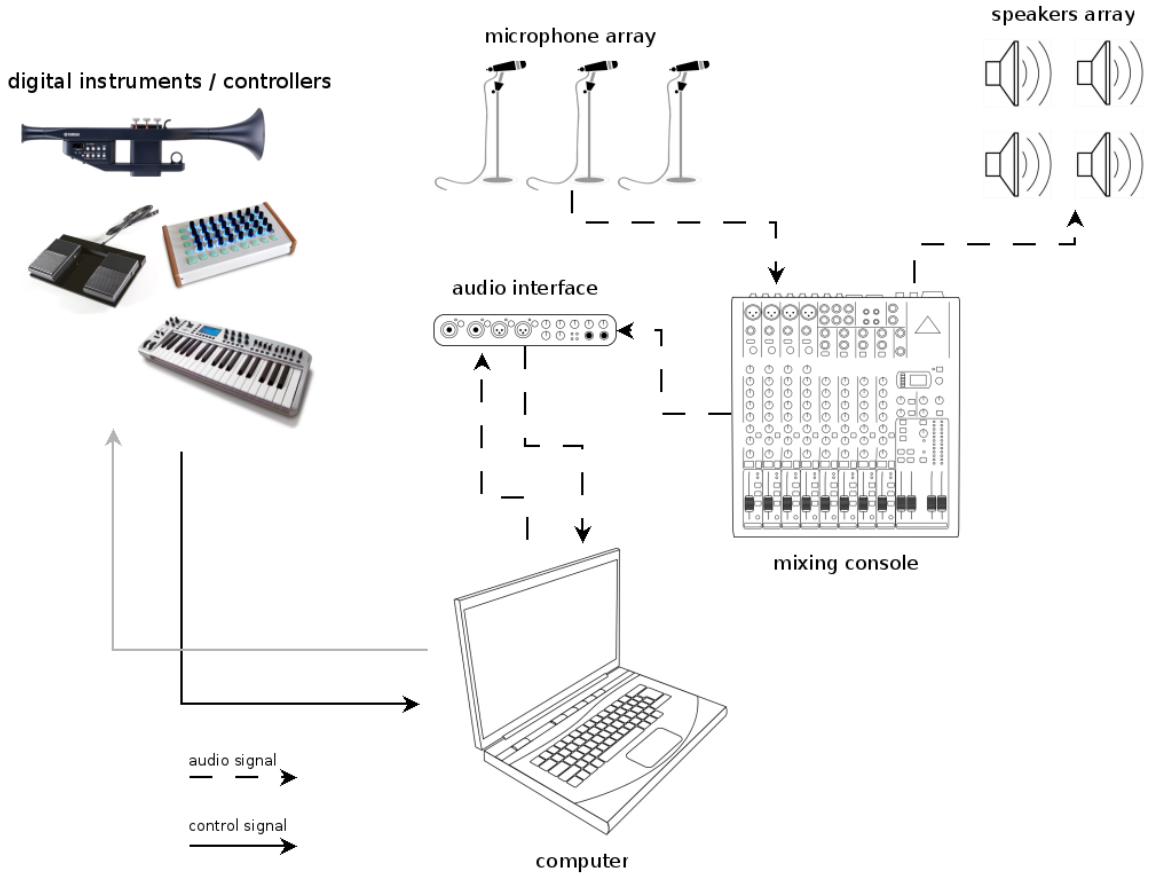


Fig. 2.7: Typical live-electronics system hardware setup

converter (ADC). Custom made devices usually use an external ADC such as the arduino boards[9](Banzi, 2009) or equivalent. In some situations the computer may be setup to send back information to the controllers in order to affect configuration or change preset during performance or rehearsal.

2.3.7.2 Live-electronics System Software

The live-electronics system software is responsible for the logical and audio operations that must occur during the musical performance, hence it must contain all information for controlling the global and local control structures (explained in sections 2.3.3 and 2.3.4) and computing the designed audio processes.

The design and implementation of the live-electronics system software has a direct im-

[9]http://arduino.cc/ last checked: 2014-09-07

pact on the final musical product. It affects both the composition and performance. Victor Lazzarini, reflecting upon the work *Mutations* by Jean-Claude Risset, goes as far as to say that it is impossible to distinguish a line between software development and music composition (Lazzarini, 2004). Horacio Vaggione also corroborates with this idea, and defends that even the programming paradigm affects the composition process. In his work "A Note on Object-Based Composition" (Vaggione, 1991), he discusses how the use of object-oriented programming paradigm enhances the connection between the composition and its implementation.

> The importance of this kind of connectivity between an object-oriented sonic process and a related compositional process cannot be overemphasized. Polymorphism is indeed a property implicitly present in any musical form of whatever style, as a dynamic element driving musical relationships, as well as their perception (Vaggione, 1991).

Computer music software can be categorized into two main branches: software utilities and programming languages. The first describes software applications that are developed to fulfill generic requirements and functional necessities that are common to a great number and wide range of users. A typical example of a software utility is the multi-track editing and mixing software that is present in most music producing studios. The latter does not relate to a software application itself, but rather it refers to the collection of instructions and rules that can be used to create software utilities and applications (Scaletti, 2002).

In the mixed music context (as previously discussed in sections 2.1 and 2.3.1), a typical live-electronics system is implemented using computer music programming languages rather than pre-existent software utilities.

In the following subsections the most mature and typically used computer music programming languages are presented, as well as the general architecture and flow of information on a live-electronics system software built with these languages.

2.3.7.3 Computer Music Programming Languages

Computer programming languages range from low-level to high-level. This discrepancy has to do with the way the user has to deal with machine specific instructions when constructing the code for a determined software. A low-level language allows the user to directly dictate instructions to the computer's central processing unit (CPU) and access specific chunks of memory. A high-level language, presents an abstraction layer that interfaces a simpler and more user-friendly code syntax with the machine code.

In that respect, current mature computer music programming languages are high-level, and built on top of more generic and low-level languages such as C, C++ and Java [10]. James MacCartney, the developer of Supercollider (described in section 2.3.7.5) justifies the higher level of abstraction that a computer music programming language requires:

> A computer language presents an abstract model of computation that allows one to write a program without worrying about details that are not relevant to the problem domain of the program. The greater the power of abstraction of a language, the more the programmer can focus only on the problem and less on satisfying the constraints and limitations of the language's abstractions and the computer's hardware (McCartney, 2002).

There are currently several computer music languages that are considered mature and well-established due to their development status[11]and their ability to achieve a wide range of sonic and compositional objectives. Despite the fact that each language presents unique features, advantages and disadvantages that are specific of their construction, there are several characteristics that are common to most of them. These common characteristics are a consequence of the fact that current audio programming languages evolved from the initial ideas presented on the MUSIC N family of computer music software and also from the methodologies derived from computer engineering and sciences.

[10]It is important to notice that the level of abstraction of a programming language is a relative classification. For instance, the C programming language might be regarded as low-level when comparing it to Java, but can be considered high-level if compared to Assembly.

The most important aspects that are present in most of the mature audio programming languages are:

- Modularity:

  Individual components of the software that present unique functionality are constructed as modules that can be exchanged and instantiated. In the audio programming domain, audio processing modules are commonly referred as unity generators (UGens), as denoted by Max Mathews in his Music N software family.

- Separation of audio and control streams:

  In order to convert the analog sound into an audio stream, the audio interface must be fed with a continuous audio stream of pre-determined size on a regular pre-determined period of time, i.e audio rate. On the other hand, a control stream, which can be constituted of several different types of control such as external MIDI values, internal feature tracking values, among others, does not present a regular periodic occurrence.

  Thus in order to process the output of an operation between a control value and an audio stream, their computation must be dealt with separately, and the operations between them occur in regular time periods (Puckette, 2006).

Current computer music programming languages can be characterized by their programming environment paradigm. The traditional paradigm is the text-based programming, where the user designs and inputs the code using text format. The more recent and increasingly more popular paradigm is the graphical programming, which allows the user to design and input code by connecting visual representations (abstractions) of logical objects and audio processes.

Text-based programming is more efficient for dealing with high complexity logical structures that need a detailed control of the order and behavior of each computation. Graphical programming languages are more efficient for rapid prototyping and creating complex

[11] A programming language is considered to be on a mature development status if it presents stable and maintained releases, an active community of users, professional and academic recognition, large portfolio of concrete projects.

audio networks from a higher level. If the audio processes are already resolved and encapsulated, it becomes easy to patch them together into a signal chain. It is important to highlight that any audio application can be programmed with both paradigms (Wang and Cook, 2008).

2.3.7.4 Max and Pd

Max is one of the most popular and widely used programming environment for computer music purposes. Named in homage to Max Mathews, Max was developed by Miller Puckette in the period of 1980-1990, at the MIT Experimental Music Studio in the early 1980's and later at IRCAM. Max was the first computer music programming environment to present the graphical programming paradigm.

Max evolved rapidly and became what is now a very stable and mature commercial programming environment owned by the Cycling'74 company. The Max paradigm, a common way of referring to the graphical programming paradigm in the computer music domain, spread out and is now present in several programming environments. The most mature and popular being the Pure Data (Pd), and Jmax. The former is a very stable and mature version of the original Max programming environment developed by Miller Puckette, the later is the Java based Max version maintained at IRCAM (Puckette, 2002).

Despite the fact that the Max paradigm is considered to be of very high level programming, Max, Pd and Jmax allows the user to create their own processing blocks (objects) using low level languages such as C, C++ and Java.

Figure 2.8 displays a simple FM synthesizer programmed with the Max programming environment. The "code" is also the user interface, which gives the user the ability to instantaneously design and control the programmed application. On the other hand, the not so clear distinction between audio process, logic operations and user interface makes it difficult to apply modern software architecture methodologies such as the Model-View-Controller (MVC) (Reenskaug, 1979), or object-oriented programming (Rentsch, 1982). Recently the Jamoma project has developed a MVC framework for Cycling'74 Max in order to deal with the lack of programming methodology (Lossius et al., 2014).

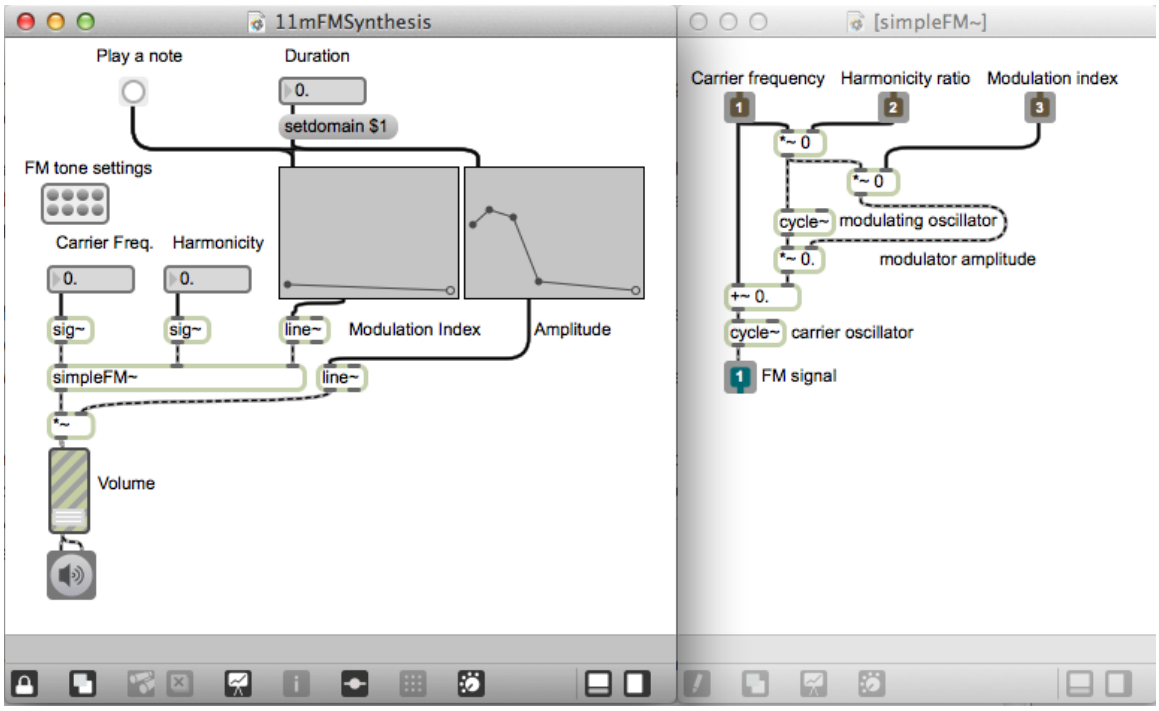


Fig. 2.8: Example of a simple Max patch for FM synthesis.

2.3.7.5 Csound and Supercollider

Csound is the current computer music programming language that still presents a very close resemblance to the Music N software family. Developed by Barry Vercoe at MIT in 1985, Csound is the most famous and long lasting text-based computer music programming language and environment. It is written in the C language and uses the concept that the composer designs instruments, orchestras and scores, each with its own individual syntax. Initially designed for non real-time operations, meaning that the finalized composition had to go through a rendering stage, current versions allow for real-time computation and control (Roads, 1996).

Figure 2.9 displays the current Csound programming environment. The main window displays the code that is being computed and the floating window accounts for the graphical user interface, enabling real-time control of parameters.

Supercollider is another example of a well established and mature computer music programming language. Developed by James McCartney in 1990, Supercollider is a text-based programming language and environment designed to deliver very efficient audio applica-

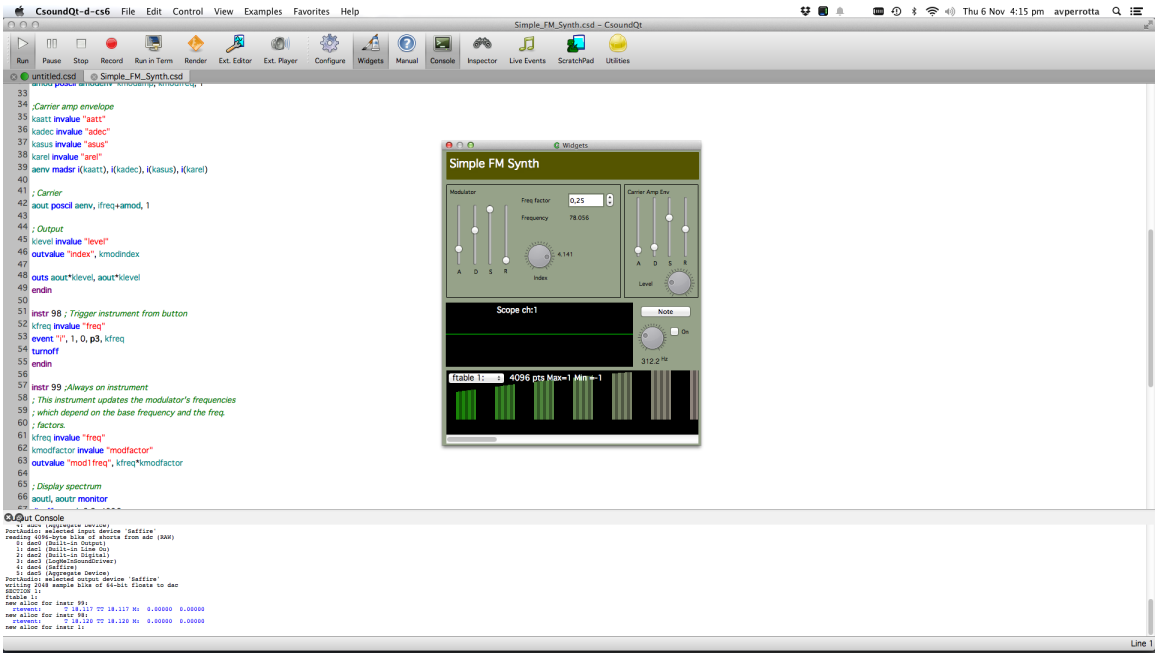


Fig. 2.9: Current Csound programming environment.

tions. It's syntax is similar to that of the very modern Smalltalk[12]programming language and it is strongly aimed towards object-oriented programming (McCartney, 1996, 2002).

Figure 2.10 displays the Supercollider programming environment. The main window displays the code that is being computed and the floating window accounts for the graphical user interface, enabling real-time control of parameters.

2.3.7.6 Generic live-electronics System Software Architecture

A typical live-electronics system software implementation presents a set of common characteristics that are independent of the used programming paradigm, language and technicalities such as operational system and choice of hardware.

Figure 2.11 displays the typical elements of an implemented live-electronics system software. The digital performer is responsible for controlling the overall progress of the events and manual parameter controls. As discussed in sections 2.3.4 and 2.3.3, this role is played by the composer, an specialized musical assistant, the conductor or the musicians. The interaction is carried on directly through the software graphical user interface, which

[12]www.smalltalk.org

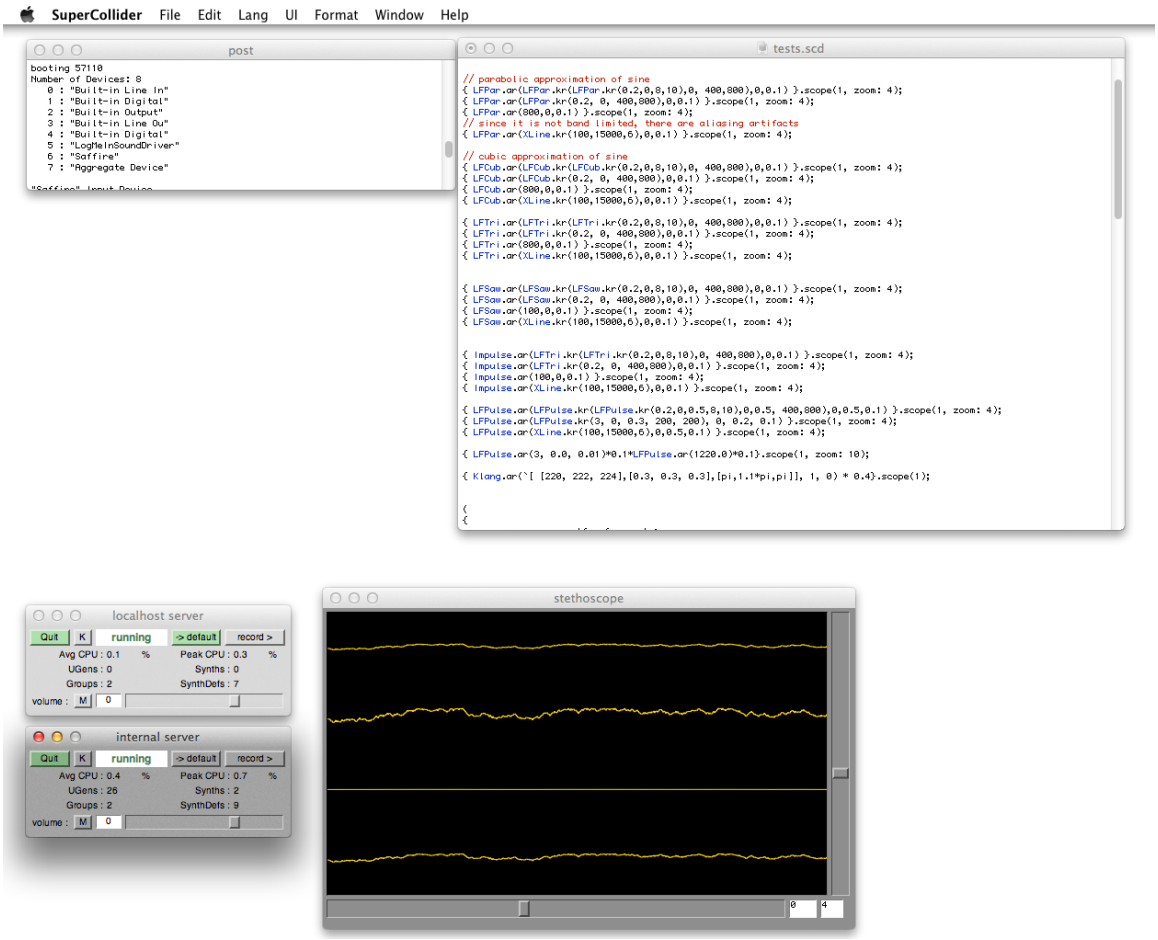


Fig. 2.10: Supercollider programming environment.

presents the most relevant control elements, or by using external physical controllers.

Each event triggers specific commands that controls the algorithmic processes, external output controls relevant for the scenography and other performance elements such as lights, video, video subtitles, and also parameter messages for the audio processes specific controls.

The audio processes are interconnected by a virtual patch bay, where the digital signal processing network is described. The patch bay assigns the path of each audio process input and output. For example, it might assign mic input 1 to go through a gain stage and from the gain stage into a reverb before outputting through the spatialization module. This virtual patch bay can be hard-coded (implemented directly in the software code in a fixed way, whereupon changes can only be made by rewriting relevant parts of the code) or in a more elegant and efficient fashion, it can be dynamically accessed and modified by the user in

real-time.

Different control signals will need to pass through specific parameter mapping stages in order to comply their output values to the required expected input values of the next element.

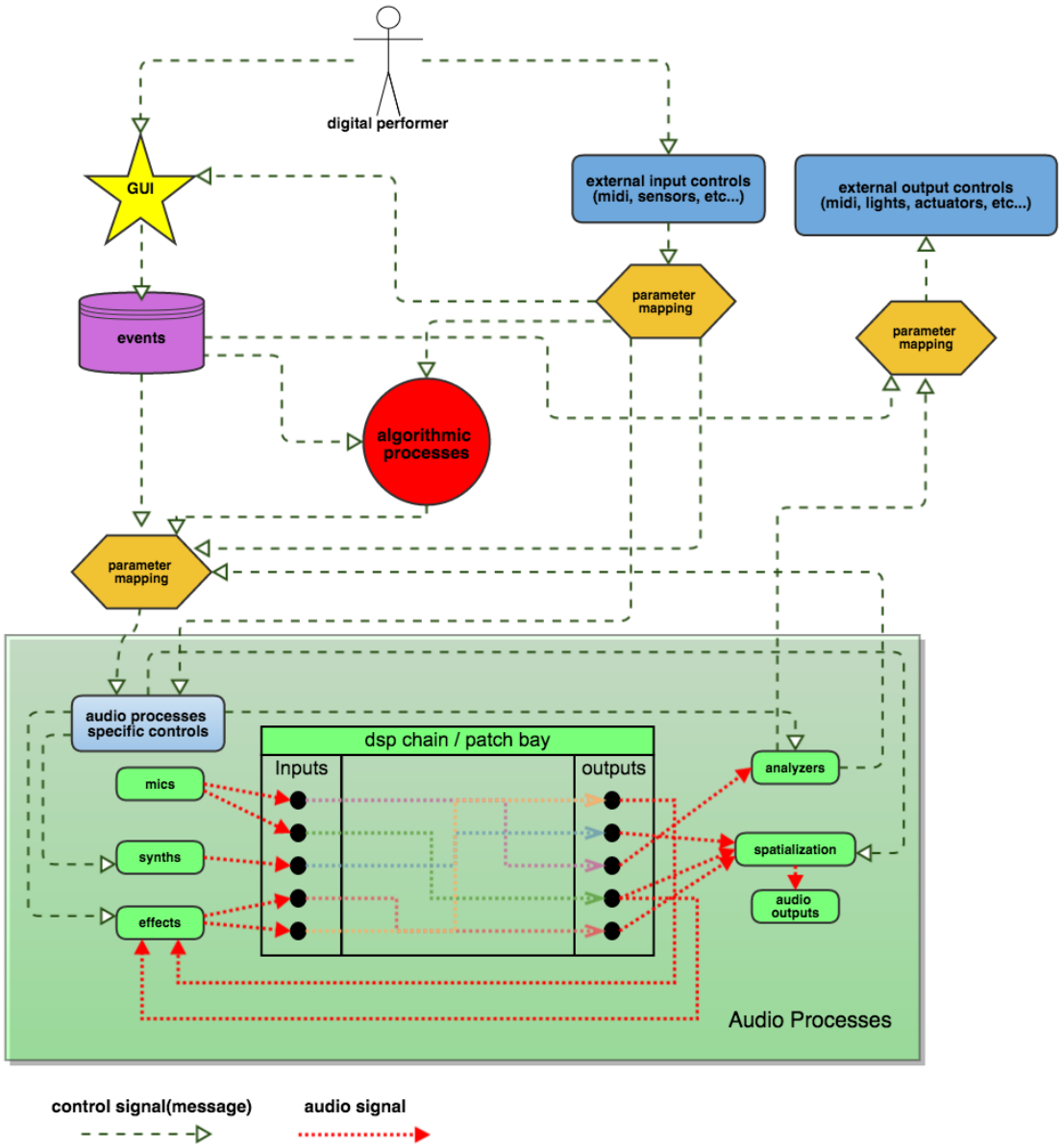


Fig. 2.11: live-electronics system software implementation diagram

Optimally the live-electronics system software implementation is highly modular, and

the GUI is simple and organized. The Pd implementation of the work *Pluton* by Philippe Manoury, implemented by Miller Puckette is a good example of a highly modular and well constructed live-electronics system software. The GUI presents all relevant elements for controlling the piece during the performance, and also accessing each module (and sub-modules) for asserting manual control of parameters. Figure 2.12 displays the GUI of the Pd patch for *Pluton*, the the top left corner displays the most relevant controls that are used during the performance, all relevant logical and audio modules are encapsulated and well organized for quick access.

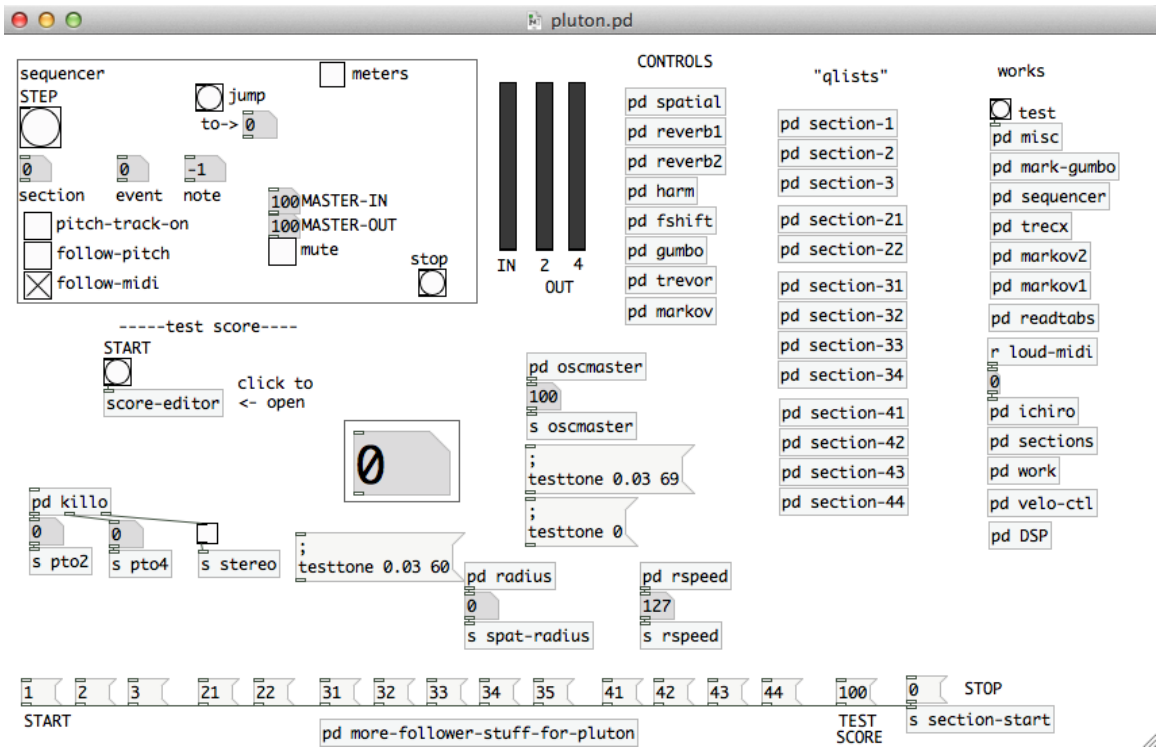


Fig. 2.12: Miller Puckette's Pd implemetation of *Pluton* by Phillipe Manoury

Chapter 3

Particle Systems

3.1 Models and Metaphors

A scientific model is a simplified representation of an object or phenomena such as physical, chemical, biological, social and philosophical processes. A model can be delivered in several distinct ways. A model can be a physical object, a fictional object, equations, descriptions, theoretical structures, and also combination of all of these (Frigg and Hartmann, 2009).

Models are an intrinsic part of our understanding of our world. We routinely think of things that surround us in terms of their models. The traffic jam flows like a very viscous fluid, the DNA is an astonishing looking colorful double helix spiral, we zoom in to the smallest bits of the fabric of matter and see tiny marbles bouncing and smashing each other, we zoom out and we see huge rock marbles orbiting around dark vortexes.

It is impossible to visualize our thoughts of the world that surround us and also the world that lies inside of us without creating metaphors that help us digest all that is intrinsically abstract and counter intuitive.

> What most distinguishes humans from other creatures is our ability to create and manipulate a wide variety of symbolic representations. This capacity enables us to transmit information from one generation to another, making culture possible, and to learn vast amounts without having direct experience... (DeLoache,

2005, p. 73)

Models play a central role in the scientific thought. They serve as an intuitive and "palpable" substitute for a target system or object (from here on referred as *target*) of observation. In that sense a model is a symbolic representation of the target; a metaphor from which it is possible to extract the most significant underlying concepts of the target and present it in a simplified way (Vorms, 2011).

Different scientific contexts and targets require different models, thus there are no rules nor recipes for guiding its construction. Consequently, the mere action of building a model serves as the learning ground for understanding the target. We learn about the target and the model not by looking at it, but rather exploring, sculpting and manipulating it (Frigg and Hartmann, 2009).

In the scope of this thesis the focus lies on computational models, more specifically on particle-based computational models. These are characterized as a class of simulation models where the target phenomena is represented by a finite number of elements (particles), each with a set of attributes that regulates their behavior and control their evolution in time (Hockney and Eastwood, 2010).

It is possible to understand a particle system as an implemented simulation of a particle-based model. Given that the focus of our study lies in computational models, the terms particle-based model and particle system are interchangeable.

3.2 Background

Particle-based modelling techniques have been part of the *computational physics* repertory since the 1940s. In this domain, particle-based modelling is used to simulate complex physical phenomena that can be represented using appropriate mathematical models, which in turn need to be discretized in order to be translated into computer algorithms. In computational physics, particle models are most used for evolutionary computations. Where given the initial state of the system and the boundary rules, it is possible to predict by simulation any future state of the system. In that matter, particle models are suitable for describing any

system that is corpuscular by nature such as fluid dynamics, plasma physics, among others (Hockney and Eastwood, 2010).

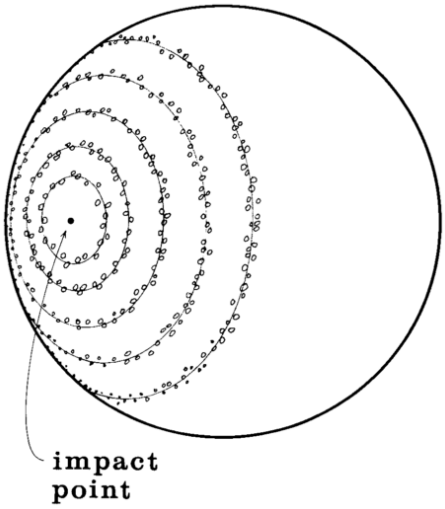
Another field that thrives with the use of particle-based models is *computer graphics*. In the computer graphics domain, the particle-based modelling technique was first formalized by William T. Reeves (Reeves, 1983), who at that time worked at the world famous Lucasfilm studios. In his work, Reeves introduced particle systems as a technique for modelling "fuzzy" objects such as fire, clouds, water and smoke. He describes fuzzy objects as a class of objects that present irregular, dynamic and complex surfaces, thus the available computer image synthesis techniques of that time were very ineffective in dealing with this class of objects. The first results of Reeeves work are shown in the *Genesis Demo* sequence from the movie *Star Trek II: The Wrath of Khan* produced at Lucasfilm Ltd (figure 3.1).

Reeves work focused on the description of complex objects and the way to synthesize their image. Nevertheless, his formalization of the particle system modelling technique opened the path for using particle systems in many other ways. In 1987, Craig W. Reynolds published a work describing the use of particle systems to model the behavior of a flock of birds (Reynolds, 1987). In his work, he used particle systems to overcome the problems of describing and controlling the motion of a large number of objects that present cohesive group behavior, but each with its own individual movement.

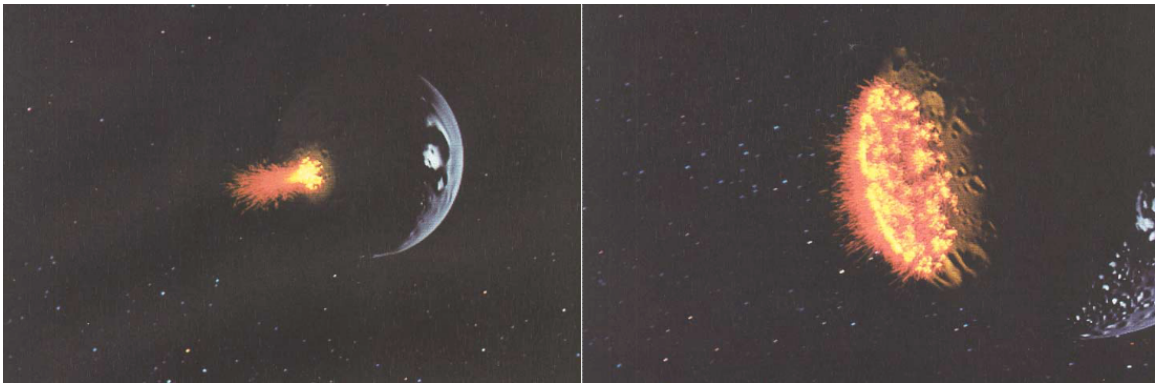
Reynolds further generalized the particle systems model formalized by Reeves by introducing the concept of *boid flock model*. While Reeves used dot-like particles to generate a complex object, Reynolds used objects composed of geometric shapes to symbolize each particle, what he called boids. Therefore he could describe a collective behavior for the entire flock (particle system) as well as singular ones for each boid (particle).

3.2.1 Particle Systems in Computer Music

In the scope of electroacoustic music, the most prominent work that applies particle-based modelling for composing and implementing electronic music has been developed by Curtis Roads (Roads, 2004). Roads work was inspired by the Nobel laureate physicist Dennis Gabor, which in 1940 proposed that any sound could be decomposed into acoustic grains



(a) Distribution of particle systems on the planet's surface



(b) Initial explosion and expanding wall of fire

Fig. 3.1: Images from *Genesis Demo* sequence from the movie *Star Trek II: The Wrath of Khan* showing the particle systems strategy at (a) and the rendered result in (b).

Extracted from (Reeves, 1983). Permission to copy without fee all or part of this material is granted provided that the copies are not made or distributed for direct commercial advantage, the ACM copyright notice and the title of the publication and its date appear, and notice is given that copying is by permission of the Association for Computing Machinery. To copy otherwise, or to republish, requires a fee and/or specific permission. © 1983 ACM 0730-1301/83/0400-0091 $00.75

representing units of time and frequency. Using Gabor's work as a starting point, Roads developed a theory focused on the exploration of acoustic events that present a duration smaller than 100ms. Roads referred to this as *microacoustical phenomena*, and the embedded time scale, which lies in the boundary of the auditory perception, as *microsound*.

The idea of creating complex sound by a collection of individual sound objects and accessing them with musical control to form a composition has been explored by several

composers. The first formalization of such strategy was proposed by Iannis Xenakis. In his 1960 article entitled "Elements of Stochastic Music" (Xenakis, 1960), he enunciates the following hypothesis:

> All sounds represent an integration of corpuscles, of elementary acoustic particles, of sound quanta. Each of these elementary particles possesses a double nature: the frequency and the intensity (the life-time of each corpuscle being minimum and invariable). Every sound, every even continuous variation of sound is to be understood as an assembly of a sufficient number of elementary particles being disposed adequately within the time level. Thus any sound complex can be analyzed into series of pure sinusoidal tones, even if the variations of these latter are infinitely close together, of short duration and complex... (Xenakis, 1960, pp. 86)

Xenakis proposition was indeed "an intuitive approach", but also a theoretical foundation that he put to practice (or through experimental validation) in his compositions. His mixed music composition for string ensemble and tape *Analogique A* (1958) and *Analogique B* (1959) are an example of truly granular compositions, where he experimented and tested his hypothesis (Solomos, 2006; Xenakis, 1992).

For Xenakis, as well as for other prominent composers, the granular paradigm relates deeply with their composition process, in which the desire to explore a wider range of sound densities and rhythmic patterns was evident. This can be observed in early electroacoustic music works such as Stochkauzen's *Gruppen* (1955-57), Ligeti's clockwork movements from the 1960's, among others. And also in the works of contemporary composers such as Agostino Di Scipio, Ludger Brümmer, Eduardo R. Miranda, and many others (Solomos, 2006).

On a more technical perspective, Roads is the responsible for turning the granular paradigm into granular synthesis, transcending it from the compositional/aesthetical level to an actual established and implementable computer music resource. Roads implementation of granular synthesis was inspired by the metaphor pictured by Xenakis:

> "A complex sound may be imagined as a multicolored firework in which each point of light appears and instantaneously disappears against a black sky... "
> (Xenakis, 1992, pp. 43)

Which reports to the previously discussed *conceptual mapping* (sections 2.3.6.2), where a metaphorical musical explanation becomes an implemented audio processing algorithm. It also shows an example of model and metaphor as discussed in section 3.1.

Granular synthesis involves the manipulation of a big number of sound particles for generating a rich and complex sound. Xenakis formalization and the consequent Roads implementation used stochastic processes to control the sound particles, where probability distribution functions were in charge of assigning the necessary parameter values for each sound particle. Using Roads work as a starting point, other computer music scientists proposed different approaches and strategies for controlling the behavior of the sound particles.

Eduardo R. Miranda proposed an implementation that uses *cellular automata* algorithm to control the production and behavior of the sonic particles (Miranda, 1995). Starting from a random distribution, the sonic particles would converge to an oscillatory pattern, resembling the behavior of harmonics in acoustic instruments, where a noisy and typically percussive sounding note attack converges to a sustained tone.

Tim Blackwell proposed a granular synthesis implementation that uses swarm algorithms (such as those of flocks of birds, fish, heards, etc.) to control the synthesis parameters (Blackwell and Young, 2004). Blackwell delineates a parallel between the self-organizing aspects of the swarm behavior and that of musicians playing freely improvised music. In both cases, the individuals of the flock (particles or musicians), have the ability to reconfigure themselves as a response to an unexpected input. Hence, a swarm behavior would enhance the interactivity aspect of creating music using the granular paradigm.

The granular synthesis paradigm uses particle systems in its most raw and fundamental form (grouping a collection of objects to form a more complex structure), however, particle systems can also be explored in order to model complex behaviors and operate the control over generic audio processes that require manipulation of large number of parameters.

In that front, the Interactive Swarm Orchestra project hosted at the ICST Zurich has

developed a set of open source software toolkits for creating swarm based computer music. The focus of the project is to develop tools that facilitate the implementation of flocking algorithms to control sound synthesis and spatialization parameters. The project has already achieved a mature level and several artistic projects have been developed using their tools (Bisig and Kocher, 2012; Bisig et al., 2008).

Particle systems have also been used in the computer music domain as a tool for elaborating complex spatial trajectories for sound spatialization. In 2005, David Kim-Boyle (Kim-Boyle, 2005) proposed an implementation of a 5 channel surround spatialization software that used the available particle system generator of the Max programming environment. In his implementation each particle is assigned an audio sample (with granular playback possibilities) and its panning is calculated by mapping the horizontal coordinate of each particle to the stereo panning of the front speakers and the vertical coordinate value to the stereo panning of the rear speakers.

More recently, Nuno Fonseca (Fonseca, 2013) has proposed a theoretical model that uses particle systems to render the audio resulting from a complex flow of particles, each representing an individual audio stream, through a 3D environment where virtual microphones are positioned. With this approach it is possible to render several distinct "views" of the same sonic complexity. This model tries to create a sonic analogy to the possibility of rendering a virtual 3D environment through the view point of different virtual cameras.

Even though the use of particle systems in computer music is still not as broad as it is in the computer graphics domain, it is a topic that is increasingly receiving more attention from researches. With the current offer of very powerful computers and software tools it is possible to imagine and actually compute very sophisticated models. The challenges ahead lie on the parameter mapping between the data calculated in the model and the coupled audio processing module.

In the most recent (as of the time of writing this thesis) issue of the Computer Music Journal, Jan C. Schacher, Daniel Bisig, and Philippe Kocher (Schacher et al., 2014) presents an article discussing several aspects of mapping between swarm systems and audio processing algorithms. Their work presents a comprehensive analysis of the several map-

ping layers required to relate the mathematical data output from a computer simulation to aesthetic qualities of an artwork.

Their approach is to segregate the technical and conceptual relationships that are intrinsic of a mapping function and discuss the possible relationships that can be extracted from the innate generative output data of a flocking simulation. They highlight the fact that the mapping function not only serves as a mathematical transformation between different data types, but also reflects the abstract artistic and aesthetic concepts of the work.

The segregation of conceptual and parameter mapping is also present in this thesis, however, where they focus on the specific case of flocking simulations, here a more general view of conceptual and technical aspects of mapping functions is presented (a more detailed discussion on this topic is presented in section 4.3.4.1).

3.3 Anatomy of a Particle System

A particle system is a collection of a finite number of particles confined in a determined environment. The dynamic behavior of each particle is governed by the rules and constraints of the environment, by their specific internal rules and by the interaction between particles.

The environment consists of a N dimensions coordinate system (where N can be any natural number), most commonly it is expressed in 2 or 3 dimensions in order to present a more intuitive description of the model and to facilitate its graphical representation. The rules of the environment may reflect well known rules of our physical world, such as gravity, collisions, friction, and so forth. However, given that a computer simulation of a particle systems is entirely described in a symbolic mathematical and logic code (software code), the model that is represented is free from assuming any connection with our physical reality. The designed environment and its rules may be completely fantastic and imaginative (Roads, 2004).

A single environment can contain different classes of particles and any number of particles of each class. Particles distinguishes themselves by their attributes. Each particle contains a set of general standard attributes and a set of class-specific attributes. General

standard attributes are responsible for enforcing the environment rules. Class-specific attributes are responsible for enforcing class specific behavior and rules. These are unique for each type of particle, therefrom they characterize that specific class.

Typical standard attributes are the particle position vector, status (dead/alive), age, time to live, and any other required by the intended model.

There are no typical or generic class-specific attributes, each type of particle will present a unique set of attributes that emerges from the intended model, metaphor and behavior.

Starting from an initial pre-determined status and configuration of the environment and its base rules, the system evolves by iterating time.

At each time iteration of the system the following steps take place:

Step 1: New particles are injected into the system (any number of particles can be injected, including zero (0 = no particles)). Their initial standard and class-specific attributes can be assigned with default pre-determined values or with a stochastic distribution:

$$attributeValue = defaultValue + randomFunction() \times variance \tag{3.1}$$

where $randomFunction()$ is a user defined random distribution function and *variance* is the amplitude of the random factor (usually, but not restricted to, generating values between -1.0 and 1.0).

Step 2: Particles attributes are updated. Standard attributes are updated according to the environments rules. Class-specific attributes are updated according to class specific rules.

Step 3: Particles that have an age value bigger than their time-to-live value, or that are tagged with the dead status value due to other determined rules are extinguished from the system.

The system can be configured to loop and iterate endlessly, or it might be configured to converge to a final state where no more particles are injected nor removed from the system.

Here we observe one of the fundamental characteristics of the a typical particle system: the use of a stochastic distribution to generate attribute values offers the ability to create an infinite number of configurations of the same system. Starting from an ideal or experimented configuration, it is possible to create an infinite set of contrasting configurations, ranging from very subtle and smooth, barely observable, value fluctuations, to drastic and profound changes that can lead to completely new, unexpected and unimagined configurations.

3.4 Implementation

In particle-based modelling, particles are described by a set of standard attributes common to all classes of particles in the system, and a set of class-specific attributes that are singular for an specific class. This aspect has a direct and deep relationship with some of the fundamental characteristics of the class-based object-oriented programming. In this programming paradigm, the user constructs his program by creating *objects* that holds data (attributes) and behavior (methods) specific of its kind. An *object* is an instance of a *class* , which in turn is the abstraction of a model. Classes can inherit attributes and methods of other classes, while adding unique ones.

Class-based object-oriented programming presents several other technicalities which are out of the scope of this discussion, but the sole property of structuring the software program using an hierarchy of classes, which are then instantiated as individual objects, justifies that a particle system can benefit form this implementation strategy.

A particle can be abstracted as a generic class, holding all the general standard attributes and methods. Different types of particles can be abstracted as child classes of the generic class, inheriting the generic attributes and adding class-specific ones. To better explain this strategy, an example is presented. Figure 3.2 shows a very simplified particle-based model of our world. It shows different classes of particles representing different elements (objects) of our world in a two dimensional environment, and also displays possible standard and specific attributes for different elements.

Object-oriented programming is therefore indeed the best programming strategy for de-

veloping particle systems, even the invention of this programming paradigm is related to simulation techniques. It was created to fit this purpose and thrives at it (Kindler and Krivy, 2011).

A particle system implementation presents two basic structures: the particle class hierarchy and a the particle system manager. The class hierarchy, as previously described, accounts for the representation of all the particle classes needed by our system. The particle system manager is the structure that holds an array with all the instantiated particles, iterates time, add and remove particles, and calls the general methods (functions) that are common to all particles. Figure 3.1 displays the pseudo C++ source code that implements (part) of the particle-based model described in the previous example (Figure 3.2). It shows the base generic class (Generic_Particle) with its attributes and methods, two specific classes (Human_Particle and Car_Particle) that inherits attributes from the base class and adds their own, the manager class (Particle _System_Manager) and a simple program initialization

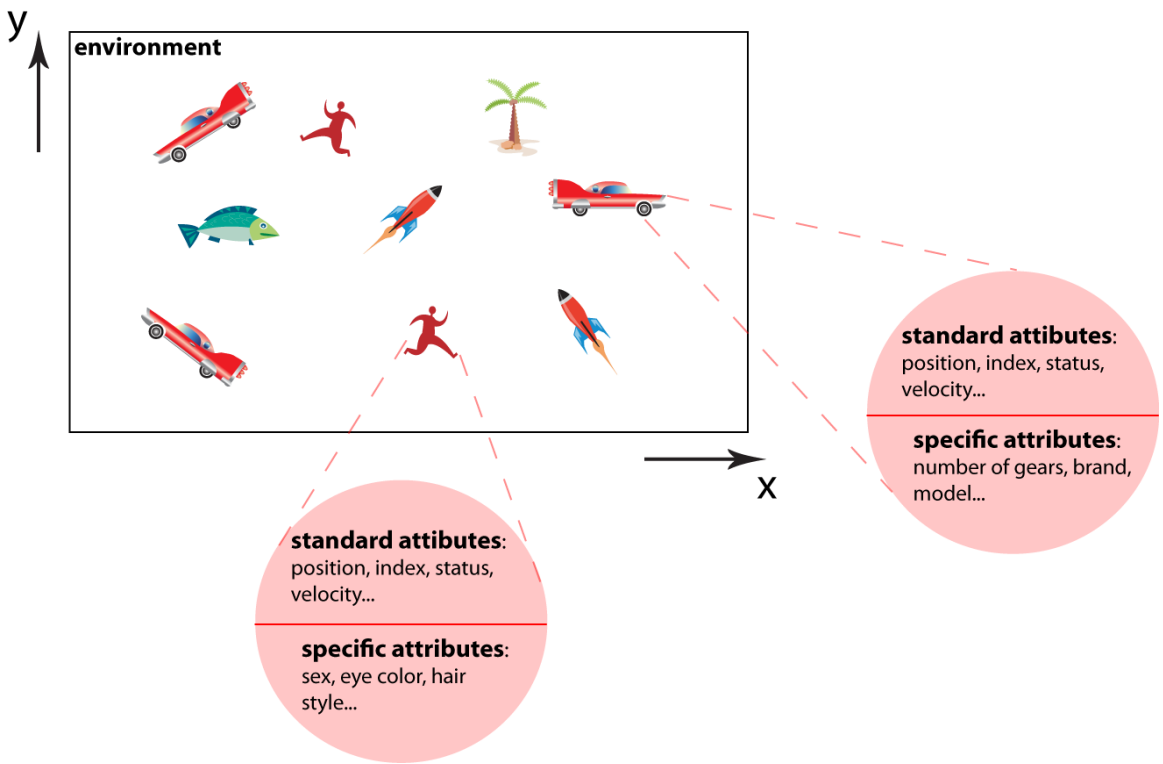


Fig. 3.2: Simplified particle-based model of our world, displaying particles general and specific attributes.

and loop (main() funciton).

The particle system manager executes two basic routines: update and drawOnScreen. This two routines can perform either in series, update and then drawOnScreen, or they can be performed in parallel, distributing each one to a different processing thread. Given that the complexity of each of this routines is drastically affected by the number of particles, the overall performance of the simulation will require a very efficient computer hardware (processor, graphics card, etc.) and software optimization in order to be able to process complex models in real-time.

```
class Generic_Particle{
    //generic attributes
    int index;
    int status;
    int age;
    float x, y;
    float vx, vy;
    //generic methods
    setup(){
        //assign initial values
        age = 0;
        status = 1; //alive
    }
    update(){
        age++;
        //move particle
        x += vx;
        y += vy;
    }
    drawOnScreen(){
        //draw the object on screen
    }
};
class Human_Particle : inherits Generic_Particle{
    //specific attributes
```

```
26      int sex;
27      color eye;
28      int hairType;
29      string nationality
30      etc...
31      //specific methods
32      blinkEye(){
33          //code for blinking eye
34      }
35      etc...
36  };
37  class Car_Particle : inherits Generic_Particle{
38      //specific attributes
39      int numberOfgears;
40      string brand;
41      string model;
42      etc...
43
44      //specific methods
45      changeGear(){
46          //code for changing gear
47      }
48      etc...
49  };
50  class Particle_System_Manager{
51      Array particles;
52      setup(){
53          //create initial system configuration
54      }
55      update(numCarParticlesToAdd, numHumanParticlesToAdd){
56
57        //add particles to the sytem
58          for(int i=0; i<numCarParticlesToAdd; i++){
59            particles.add(new Car_Particle);
60          }
```

```
61
62          for(int i=0; i<numHumanParticlesToAdd; i++){
63            particles.add(new Human_Particle);
64          }
65
66          //updates all particles and remove dead ones
67          for(int i=0; i<particles.size(); i++){
68              particle[i].update();
69              //test if particle is dead and remove it
70              if(particle[i].status == 0){
71                  remove particle[i] from particles array;
72              }
73          }
74      }
75
76      //draw particles on screen
77      draw(){
78          for(int i=0; i<particles.size(); i++){
79              drawParticlesOnScreen();
80          }
81      }
82  };
83  int main(){
84      //create a particle system manager
85      Particle_System_Manager psm;
86      psm.setup();
87      //executes the program
88      while(program is running){
89          psm.update(numCarParticlesToAdd, numHumanParticlesToAdd);
90          psm.draw();
91      }
92  }
```

Code Figure 3.1: Pseudo C++ code for the particle-based model presented in figure 3.2

In order to address this issue, different strategies may take place. The update and draw

routines can be assigned to different processing threads on the CPU, and the user might prioritize one or the other. Another possibility is to explore the current capabilities of graphical processing unities (GPU) and assign the computations required in the update routine to the GPU computation and memory bandwidth (Kipfer et al., 2004).

3.4.1 Implementation Tools

As discussed in the previous section (3.4), an efficient particle system implementation strategy benefits from the object-oriented programming paradigm. Thus any language capable of performing object-oriented programming is a good candidate. The choice of the language should be made taking into considerations the availability of pre-existing libraries and frameworks that offer the best set of tools for creating this kind of computer simulations.

The Max programming environment offers a collection of objects inside its Jitter package (Max's imaging processing and openGl rendering modules) that allows the user to construct simple particle systems. Jitter modules allow easy implementation of a 3D environment where particles can be injected at any rate and controlled using classical mechanics forces. Additionally, Jitter offers modules for creating physically correct rigid-body physics, enabling the user to add realistic collision detection between objects that preserve the classical mechanics laws such as linear and angular momentum.

The fact that Jitter modules are not open source and that the Max visual programming paradigm (see section 2.3.7.4) doesn't allow object-oriented programming, hinders the agile development of systems that present complex dynamics and multiple kinds of particles. Nevertheless, it offers an easy integration of the implemented particle system with its audio processing engine.

If the choice is to develop a particle system "from scratch" using lower level programming languages such as C++ or Java, there are several open source libraries dedicated to dynamic simulations that can help an experienced programmer getting the job done, such as Bullet Physics (Coumans, 2006), Open Dynamics Engine (Smith, 2006) and Box2D (Catto, 2010).

Chapter 4

Particle Systems Modelling in Live-electronics

In the previous chapters (2 and 3) a discussion of the definitions, concepts, state-of-art and implementation strategies for live-electronics systems and particle systems was presented. The following chapter will present the method for applying a particle-based model in a live-electronics system implementation.

A logical formalization of the a general live-electronics system is proposed and then modified to integrate a particle-based model. From this formalization it is possible to extract the possible features, advantages and disadvantages of the method.

In order to discuss the possibilities of using particle systems in the implementation and also on the elaboration of the live-electronics of a mixed music composition, it is necessary to create a logical formalization of a general live-electronics system from where it is possible to visualize which structures are suitable of receiving a particle-based model.

4.1 Logical Formalization

In the computational physics domain, the steps required for creating a computer experiment can be described as follows (Hockney and Eastwood, 2010):

Step 1: The starting point is the physical phenomenon that will be modeled (target).

Step 2: A mathematical model for the target is proposed.

Step 3: A discrete algebraic approximation of the mathematical model is performed.

Step 4: The algebraic approximation is programmed into a computer algorithm.

Step 5: The simulation program containing the algorithms is implemented.

Step 6: The simulation performs the computer experiment.

Figure 4.1 displays the enumerated steps in a diagram format.

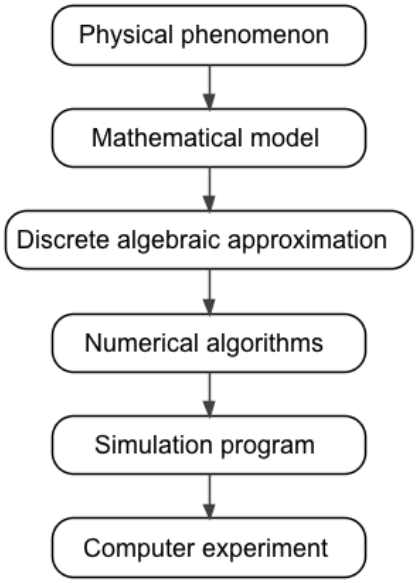


Fig. 4.1: Logical flow of the required steps for a model based computer experiment.

Based on this workflow, it is possible to describe an analogous process chain for modelling and implementing a live-electronics system based on a compositional and aesthetic phenomenon:

Step 1: The starting point is the compositional and aesthetic ideas.

Step 2: The composition is broken down into unique sonority states (defined in section 4.2).

Step 3: Each sonority state is modeled by an audio processing chain and a parameter control structure.

Step 4: The audio processes and parameter control functions are implemented as modular algorithms.

Step 5: The overall live-electronics system is implemented.

Step 6: The system performs in a real musical performance.

Figure 4.2 displays the enumerated steps in a diagram format.

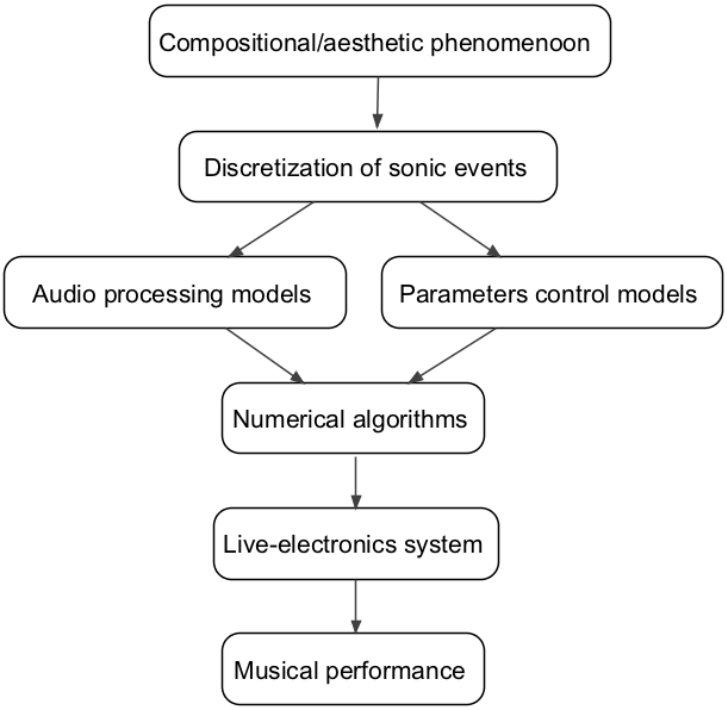


Fig. 4.2: Logical steps for implementing the live-electronics of a mixed music composition.

In this workflow, step 2 represents a mathematical formalization of the live-electronics system and step 3 represents the appropriate modelling of the assets and functions that emerge from the formalization and in fact implement and control the dynamic behavior of the sonic events.

This logical process is also supported by Xenakis description of the fundamental phases of a musical work. He proposes that starting from the initial concepts and intuitions, the following step is the definition of sonic entities, their symbolism and connection with conceivable sound sources. The subsequent step is to describe the transformations that each

sonic entity must perform during the course of the composition (i.e. dynamic behavior of the sonic events) (Xenakis, 1992).

4.2 General Formalization of the Live-electronics System

The first step towards a formalization of the live-electronics system is to create a discrete segregation of all sonic events that need to occur during the composition and identify each event state with the respective audio processes and parameter configuration that characterizes that state.

We define a unique configuration of audio processes including their respective control parameters as a *sonority state*. The conceptual transformations are translated into *sonority state transitions*. Thus we define:

A sonority state at any state i:

$$S_i \tag{4.1}$$

The live-electronics of a mixed music composition Le can be discretized into N states S_i that represent each sonority state.

$$Le = \{S_1, S_2, \ldots, S_N\}, \forall N \in \mathbb{N}, N > 0 \tag{4.2}$$

Each sonority state S_i is implemented by an audio processing network that is itself composed of several distinct audio processing modules organized in any network configuration, performing individual processes in series or parallel. Thus we define the group of n audio processes that implement S_i:

$$P^{S_i} = \{p_1, p_2 \ldots, p_n\}, \forall \{n, i\} \in \mathbb{N}, n > 0, 0 < i \leq N \tag{4.3}$$

Where each p_j represents one audio process module.

Each audio process module p_j contains a set of parameter variables that need to be controlled so that the sonority state is characterized. Thus we define the group of m low-

level control variables (i.e. parameters) of each audio process p_j:

$$X^{p_j} = \{x_1, x_2 \ldots, x_m\}, \forall \{m, j\} \in \mathbb{N}, m > 0, 0 < j \leq n \tag{4.4}$$

The low-level control variables refer to the parameters that can be controlled from outside an audio process module. This definition is relevant for cases where the audio process module already contains an hierarchy of variables in its implementation. A better way to understand this is by looking at the following example (same one as presented in the parameter mapping discussion, section 2.3.6.1): A simple reverberation effect can be implemented by a network of delays with feedback. In order to control the audible acoustic qualities of the reverberation effect, such as room size and decay time, it is necessary to control the delay time and feedback values (lowest-level variables) for all the delays in the network. In this case, it is usual to create a hardcoded mapping connecting all of the lowest level variables to the actual observable parameters. Ergo, from the user perspective the low-level variables that can be accessed are not literally the lowest-level variables used in the audio process implementation.

Given two consecutive sonority states S_i and S_{i+1}, we define the continuous passage between states a sonority state transition:

$$T : S_i \rightarrow S_{i+1}, \forall i \in \mathbb{N}, 1 \leq i \leq N \tag{4.5}$$

By applying definitions 4.3 in 4.5 we obtain :

$$T : S_i \rightarrow S_{i+1} \implies P^{S_i} \rightarrow P^{S_{i+1}} \tag{4.6}$$

While the left side of 4.6 shows a conceptual transformation, the right side expresses the implemented (or implementable) transition. However, a transition can only be operated by controlling the low-level variables X (equation 4.4), thus we need to further develop definition 4.6.

Given,

$$P^{S_i} = \{p_1, p_2 \ldots, p_n\}, \forall n \in \mathbb{N}, n > 0 \tag{4.7}$$

the group of audio processes that implement sonority state S_i, and

$$P^{S_{i+1}} = \{\bar{p}_1, \bar{p}_2 \ldots, \bar{p}_{n'}\}, \forall n' \in \mathbb{N}, n' > 0 \tag{4.8}$$

the group of audio processes that implement sonority state S_{i+1}, a sonority state transition can develop in two different ways:

Case 1:

The audio processes that form the group P^{S_i} are exactly the same ones that form $P^{S_{i+1}}$, thus:

$$n = n' \tag{4.9}$$

and

$$p_j \cong \bar{p}_j \tag{4.10}$$

(It is important to highlight the fact that it is not possible to use the mathematical equality "=" in statement 4.10. This is due to the fact that the state (values) of the low-level variables X^{p_j} and $X^{\bar{p}_j}$ are not the same. Different sonority states assume different parameters configuration)

The sonority state transition can be expressed in terms of the low-level variables:

$$T : S_i \rightarrow S_{i+1} \implies P^{S_i} \rightarrow P^{S_{i+1}} \tag{4.11}$$

$$p_j \rightarrow \bar{p}_j \implies X^{p_j} \rightarrow X^{\bar{p}_j} \tag{4.12}$$

and thus it can be in fact be implemented.

Case 2:

The audio processes that form the group P^{S_i} are different from the ones that form $P^{S_{i+1}}$.

In this case, statement 4.12 is not valid, therefore the transition must be solved by creating a sub-state $\tilde{S}_i$, formed by the same audio processes that form S_i. And a sub-state $\tilde{S}_{i+1}$ formed by the same audio processes that form S_{i+1}.

$\tilde{S}_i$ represents a final state for S_i, while $\tilde{S}_{i+1}$ represents an initial state for S_{i+1}. The overall transition can be expressed as two simultaneous transitions:

$$T : S_i \rightarrow S_{i+1} \implies P^{S_i} \rightarrow P^{S_{i+1}} \implies \begin{cases} P^{S_i} \rightarrow P^{\tilde{S}_i} \\ P^{S_{i+1}} \rightarrow P^{\tilde{S}_{i+1}} \end{cases} \tag{4.13}$$

And the sub-transitions (right side of statement 4.13) can be expressed in term of the low-level variables as shown in Case 1 (statements 4.11 and 4.12).

As discussed in section 2.3, the control over the low-level variables (X^{p_i}) is part of the local control structures, and ultimately by the global control structures. Direct control over variable values is designed to be either automated or non-automated (see definition in section 2.3.4). Thus we can define:

The group of V input variables of the live-electronic system:

$$Y = \{y_1, \ldots, y_V\}, \forall V \in \mathbb{N}, V > 0 \tag{4.14}$$

Here there is no distinction on the source of the input varible y_k. It represents external inputs such as those from physical interfaces, data resultant from audio features tracking and data resultant from algorithms.

In order to couple the input variables y_k to the low level variables x_u (statement 4.4) a set of control functions need to be implemented.

For a given sonority state S_i, the respective group of audio processes $P^{S_i} = \{p_1, p_2 \ldots, p_n\}$ and the group of low-level variables of each audio process p_j, $X^{p_j} = \{x_1, x_2 \ldots, x_m\}$, each

low level variable x_u is expressed as:

$$x_u = Ctrl_u(Y,t), \forall u \in \mathbb{N}, 0 < u < m \tag{4.15}$$

where $Ctrl(Y,t)$ is a control function and t is time (absolute of the entire composition or relative to a specific event).

The definition of the control function as a function of the input variables and time is justified by the fact that it must express both automated (time dependent) and non-automated functions (no explicit time dependence).

From the control function definition it is possible to observe that it must serve two purposes: it must facilitate the technical link between different data formats and, at the same time, mathematically represent the path through which conceptual transformations are performed. This is indeed the role of the parameter mapping functions (discussed in section 2.3.6.1). In order to satisfy all possible mapping relationships (one-to-one, one-to-many, many-to-one) we further generalize the control function. Consequently each low level variable x_u is expressed as:

$$x_u = Map_u(X^{p_j}, Y, t) \tag{4.16}$$

Where $Map_u(X^{p_j}, Y, t)$ is function that maps the input variables, low-level variables and time to a specific low-level variable.

4.3 Particle System Modelling Applied to the General Live-electronics

From the general live-electronics system formalization presented in section 4.2 is is possible to apply a particle-based modelling method for describing and implementing such a system.

4.3.1 Fundamental Concept

The particle-based model for a live-electronics system is based on the fundamental concept that each audio process p_j (statement 4.3) is represented by distinct type of particle. Hence a sonority state S_i (definition 4.1) and the respective group of audio processes P^{S_i} is characterized by a specific configuration of the particle system.

As a consequence, a sonority state transition $T : S_i \rightarrow S_{i+1}$ is achieved by controlling the dynamics of each particle in the system and also the overall behavior of the system (adding/removing particles).

In order to implement this concept, the set of class-specific attributes of each type of particle is designed to correspond to the low-level variables of the reciprocate audio process.

Figure 4.3 displays an illustrative example of a particle system model for a live-electronics system, displaying the standard and class-specific attributes of different particle types.

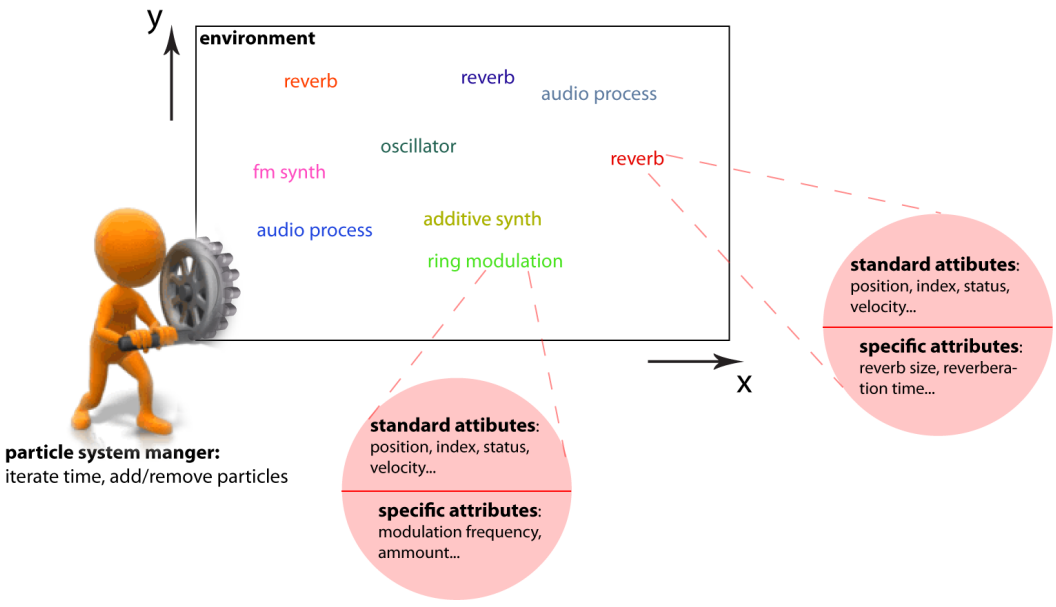


Fig. 4.3: Illustrative example of a particle system model for a live-electronics system, displaying the standard and class-specific attributes of different particle types.

4.3.2 Formalization

Based on the *audio process* $\leftrightarrow$ *particle* concept, it is possible to adapt the formalization presented in section 4.2.

We define:

A particle that represents an audio process p_j:

$$\alpha^{p_j} \tag{4.17}$$

The group of r standard attributes of any particle α:

$$A = \{a_1, a_2, \ldots a_r\}, \forall r \in \mathbb{N}, r > 0 \tag{4.18}$$

The group of Q global attributes and constants of the particle system also containing the input variables Y (definition 4.14).

$$G = \{g_1, g_2 \ldots g_Q, Y\}, \forall Q \in \mathbb{N}, Q > 0 \tag{4.19}$$

In order to implement a sonority state transition (T) by controlling the dynamics of the particles α^{p_j}, it is necessary to implement a set of mapping functions that link the low-level attributes of an audio process to the standard attributes of the respective particle. Furthermore, it is necessary to implement a set of functions that control the dynamics of the particle by operating exclusively on the standard attributes. Thus we substitute the previously defined mapping function $Map_u(X^{p_j}, Y, t)$ (statement 4.16) and define:

The function that maps a low-level variable to the standard attributes:

$$x_u^{p_j} = f_u(A), \forall k \in \mathbb{N}, 0 < u \leq m \tag{4.20}$$

The function that controls the dynamics of the particle:

$$a_s^{p_j} = h_s(G, t), \forall s \in \mathbb{N}, 0 < s \leq r \tag{4.21}$$

where t is time (absolute or relative).

Considering a sonority state transition $T : S_i \rightarrow S_{i+1}$ where the group of audio processes that implement S_i is the same that implement S_{i+1} (same case as described in 4.11 and 4.12), it is possible to state:

$$T : S_i \rightarrow S_{i+1} \implies P^{S_i} \rightarrow P^{S_{i+1}} \tag{4.22}$$

$$p_j \rightarrow \bar{p}_j \implies X^{p_j} \rightarrow X^{\bar{p}_j} \implies a_s^{p_j} \rightarrow a_s^{\bar{p}_j} \tag{4.23}$$

From statement 4.23 it is possible to observe that once the mapping function f_u that maps the standard attributes A to the low-level variables X^{p_j} of the respective audio process is established, the control over transitions is inferred by operations on the standard attributes. The sonority state is thus in fact controlled by the dynamic behavior of the particles.

4.3.3 Implementation Strategy

In order to implement the particle-based modelling method, it is necessary to divide the live-electronics system into two distinct parts; the particle system and the audio processes system. This separation is justified by the fact that each system requires a different set of programming tools. Furthermore it is a reflection of the segregation of audio and control streams already present in a typical live-electronics system implementation (discussed in section 2.3.7.3). The particle system runs at control stream rate, and the audio processing system runs at audio sample rate.

The particle-based modelling method presented in sections 4.3.1 and 4.3.2 demands that some specific implementation requirements are observed in order to achieve a functional particle system and reciprocate audio processing system.

The particle system must be developed using the object-oriented programming paradigm and the audio processing system must be implemented so that each audio process is implemented as a module, and the system architecture must allow the dynamic management of any number of instances of each module. The number of instances should be only limited

by the processing capabilities of the hardware.

One possible solution for this relies on the use of the Max programming environment for implementing the audio processing system. In Max, each audio process module can be encapsulated using the **poly~** object, which provides automatic voice allocation, voice management and individual or general voice access. Additionally the **poly~** object automatically distributes the overall processing among the CPU cores (Battenberg et al., 2010).

The integration of both systems can be achieved in two distinct ways:

- The particle system and the audio processing system are implemented as separate stand alone applications and the transfer of data between both systems is done using a suitable communication protocol.

  In this scenario, a typical solution is to use the Open Sound Control (OSC) protocol, which was developed for the specific purpose of interconnecting audio and multimedia devices and software using modern network technology (Wright et al., 2003).

  OSC offers the advantage of being a very well established and mature communication protocol that is typically embedded as a native utility in the most relevant computer music programming environments such as Max, Pd and Supercollider.

  In any case, several other protocols, such as MIDI, TCP/IP, among others, can be used to connect both systems, and the choice of protocol is entirely dependent on the requirements of the overall system and musical performance.

- The particle system and the audio processing system are implemented as one application, thus the integration is implicit inside the overall system.

  The advantage of this approach is that it avoids any limitation and latency that can be encountered when using communication protocols. Thus for systems that require a very large size of data being communicated from the particle system to the audio processing system this strategy is more efficient.

  On the other hand, an implementation that segregates the particle system and the audio processing system offers the possibility of having each system running in a separate com-

puter, thus offering the possibility of having remote control of the audio processing system and of course more "processing horse power" by using several CPUs and audio interfaces.

4.3.4 Analysis of the Formalization

The formalization of the general live-electronics system (section 4.2) and its adaptation for particle-based modelling (section 4.3.2) offers the possibility to understand and discuss several aspects of the process of developing and implementing a live-electronics system in great technical detail.

4.3.4.1 Mapping Structures Detailed

From the formalization it is possible to observe the clear distinction between the conceptual mapping layer and the parametric mapping layer discussed in section 2.3.6.

The conceptual mapping layer is explicitly evidenced in statement 4.3, where the group of audio processes (P^{S_i}) that implement a sonority (S_i) is assembled. It is the first logical step that binds the abstract compositional universe to the implementable sound generating universe. The previous step, where a discrete map of sonority states is created, is indeed a pure compositional process, which in the context of mixed music, is completely interlaced with the instrumental composition.

The act of choosing the group of audio processes that will implement a specific sonority state is analogous to the process of choosing the instruments that perform an specific passage in an orchestration process. It is the first step towards the extraction of a specific timbre, that will be further polished to the final desired result by controlling the available parameters and dynamic transitions.

The conceptual mapping is also present, but now implicitly, in the control functions ($Ctrl_u(Y,t)$, statement 4.15) and consequently in its generalization into mapping functions ($Map_u(X^{p_j},Y,t)$, statement 4.16).

A mapping function in its simplest form is responsible for converting the format of the incoming data to a format that the receiver can interpret. A simple example would be a scaling function that converts the range of values. If only looked through this perspective,

the mapping function looses its conceptual relationship with the compositional ideas, it becomes just another technical block of the system.

However, there are cases where the mapping function presents a direct relationship with the composition and also the musical performance. In this cases it is harder to see the separation between the technical part (parameter mapping) and the conceptual part embedded in the function.

For the purpose of understanding the limits and roles of the parametric and conceptual mappings a detailed analysis of possible sonority state transitions is proposed:

Assuming a hypothetical sonority state transition:

$$T : S_i \rightarrow S_{i+1} \tag{4.24}$$

where the sonority states S_i and S_{i+1} are implemented by the same group of audio processes that contain only one audio process:

$$P^{S_i} \cong P^{S_{i+1}} = \{p_1\} \tag{4.25}$$

and the group of low-level variables of p_1 contains only one variable:

$$X^{p_1} = \{x_1\} \tag{4.26}$$

In this scenario, the transition can be expressed as:

$$T : S_i \rightarrow S_{i+1} \implies P^{S_i} \rightarrow P^{S_{i+1}} \implies {x_1}^{S_i} \rightarrow {x_1}^{S_{i+1}} \tag{4.27}$$

The actual control over x_1 is implemented by the mapping function, thus:

$${x_1}^{S_{i+1}} = Map({x_1}^{S_i}, Y, t) \tag{4.28}$$

The number of possible mapping functions is infinite. However, it is possible to separate them into two distinct groups: time dependent mapping functions and non-time depen-

dent[1] mapping functions.

If the mapping function is non-time dependent, the transition from the initial value ${x_1}^{S_i}$ to the final value ${x_1}^{S_{i+1}}$ is performed by an external input. If we overlook the case where the transition is discrete (non continuous), meaning that the value ${x_1}^{S_{i+1}}$ is triggered by a simple event detection (automated or not), the only possible way to express the transition is by a non-automated external control (previously discussed in section 2.3.4).

In this case, the mapping function is predominantly technical. The transition can only be performed by a synchronous bond between the external input and the low-level variables. This falls in the realm of the very well established and discussed parameter mapping topic (previously discussed in section 2.3.6.1). The criteria for designing the mapping function is affected by the relationship between the external control and the audio process that follows that control. The morphology of the transition is not expressed in the mapping function itself, instead it is performed by the digital performer or instrumentist (depending on the type of external control). The musician or digital performer are directly responsible for the expressiveness and precision of the transition. The mapping function is a technical tool, and if well designed it is transparent to the performer.

In the case that the mapping function is time dependent, the transition must be performed by the morphology of the designed function. The expressiveness and precision of the transition is thus a direct consequence of the implemented algorithm. It conceals an implicit conceptual relationship with the compositional ideas, and the conceptual mapping (i.e. the act of translating the abstract musical and aesthetic ideas into implementation) becomes the predominant aspect of the function.

4.3.4.2 Modelling vs Mapping

The most evident feature provided by the use of the proposed particle-based modelling technique for a live-electronics system is the explicit separation between the parameter mapping functions and the functions that control the dynamic behavior of the system, and conse-

[1] A function $f(v,t)$ where v is an any dimension vector and t is time, can still be non-time dependent, for example: $f(v,t) = av + 0t$.

quently the morphology of the sonority state transitions. This is observed in statements 4.20 and 4.21 respectively.

The addition of a new set of functions (h_s) to control the configuration and dynamic behavior of a sonority state presents a particular characteristic. By operating in h_s, the user always deals with the same standard attributes, regardless of the audio process that will be affected. Even in the case where the audio processes are unknown, it is still possible to design the algorithms and functions that represent the desired dynamic behavior. This is in fact the act of creating a model, and as discussed in section 3.1, the model serves as a test ground for experimenting different audio process and configuration of audio process until the desired result is achieved.

Thereupon, the function h_s assumes the role of modelling function, and its biggest distinction to a mapping function is the fact that in order to design a mapping function, the input source of the incoming data and the output destination of the transformed output data must be known before the design process begin.

4.3.4.3 Micromodulations and Layering

Audio layering is a common strategy used for achieving a rich and complex spectrum from a simpler audio source. The strategy dictates that several instances of the same audio are superimposed and each instance receives a subtle deviation in its dynamic envelope (both in the time and amplitude domains). The result is a thicker version of the original sound with enriched perception of harmonic content (Viers, 2011).

This technique is commonly used in the sound design domain. In this context, given that the final sonic output need not to be achieved in real time and the source material is available in the form of audio samples, the designer has the possibility to manipulate each sample with surgical precision (using current digital audio workstations) until the desired result is achieved.

In the real-time computer music domain, this technique is usually neglected due to the fact that handling several instances of an audio process, specially if it is a process that transforms the live instrumental input, is very CPU consuming and the number of parameters

that need to be controlled in real time is more often than not overwhelming (Risset, 1999).

Using particle systems to control the injection and removal of new particles/audio processes of the same type in the system using a stochastic distribution with very small variance in the initial and constraint values of each particle is a suitable solution for generating an audio layering effect with any kind of audio process.

By mapping the time-to-live attribute of the particles (how many iterations of the system a particle can participate before it is removed) to the output amplitude (volume) of their respective audio process, it is possible to achieve distinct forms of superposition just by controlling the rate that particles are injected into the system and the randomic variance of their time-to-live attribute. Figure 4.4 shows the layering of 3 instances of the same audio process. The layering is achieved by injecting 3 particles into the system with a very short time interval between the injection of each particle. For ease of understanding, the randomic variance of attribute values is very small (non-observable).

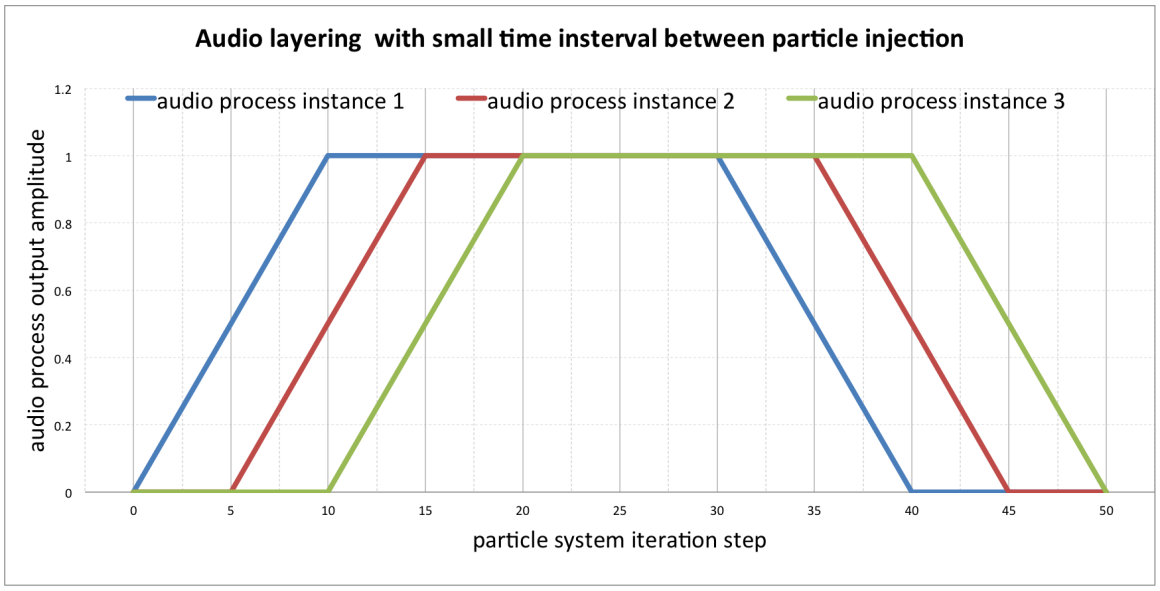


Fig. 4.4: Audio layering using small time interval between particle injection and very small (non-observable) randomic variance for attribute values

By controlling the particle injection rate and the randomic variance of particles attributes, several other layering effects can be achieved. This control allows the user to create a continuous evolution of the same sound, with micromodulations that are cross-faded over

time. This is achieved by carefully adjusting the particle injection and the particles time-to-live default and randomized values. Figure 4.5 displays the layering of 3 instances of the same audio process that present small modulation and are cross-faded.

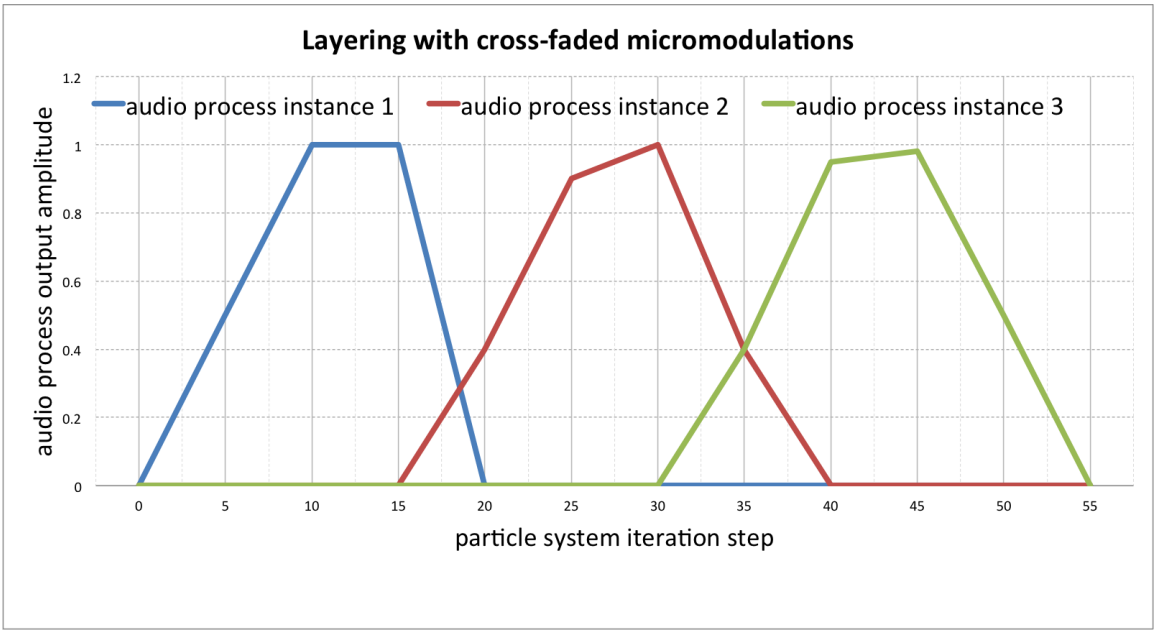


Fig. 4.5: Audio layering using cross-faded micromodulated instances of the same audio process micromodulations

4.3.4.4 Stochastic Transitions

Sonority state transitions that are time dependent require a control function that is also time dependent in order to perform a desired transition morphology. This has aesthetic consequences that may be undesirable by the composer such as glissandi between different frequencies, periodic repetition due to feedback, among others.

In order to avoid this kind of unwanted effects the control function must be non-continuous. However, this can result in a non-continuous sonic output during the transition. Which in turn might also be an unwanted effect.

To better understand this problem, an example is presented: Consider the case that the initial sonority state is implemented by an additive synthesis configured with a specific harmony. The consecutive sonority state is implemented by the same additive synthesis

process, but configured with a different harmonic spectrum. If the transition is carried on by a continuous time dependent function, the acoustic perception will be one of a glissando between the involved frequencies. If the transition is carried on with a non-continuous function, the acoustic perception will be one of a sudden change of harmony.

If the desired effect is to achieve a continuous change of harmony, without sudden harmonic changes, the typical solution is to implement the initial and final sonority states using distinct instances of the additive synthesis process and cross-fade the amplitude of both instances.

Using the principle of audio layering and micromodulations discussed in the previous section (section 4.3.4.3), it is possible to use stochastic transitions to solve the afore mentioned transition problem. By controlling the injection of particles in the system with a probability distribution that changes over time, favoring one type of particle in the beginning and migrating to the second type of particle towards the end of the transition, it is possible to achieve a superposition of the initial and final sonority states, and any other desired sub-states (or micromodulations) that lies in the middle. The transition is then performed just by controlling the probability of injecting the initial or final state particles in the system, the initial values and randomic variance of the particles attributes.

4.3.4.5 Spatialization

Space is an intrinsic aspect of sound. Given that sound is a mechanical phenomenon, therefore it must occupy a space. As a consequence, space is also an inseparable aspect of music, regardless of the source (instrumental or electronic), and regardless if the spatial attribute of the musical sound is explicitly manipulated or not (Menezes, 2006).

Sound spatialization is segmented in two different opposites, declared by Michel Chion as: the internal space and the external space (Chion, 1988). The former refers to the conceptual manipulation of the spatial attributes of the musical sounds. It is the compositional aspect of space, which in electronic generated music can be profoundly manipulated. Aspects such as sound position, temporal trajectories, spatial distance between sounds, presence and absence of sounds can be easily accessed and controlled with the current audio

programming tools.

The latter refers to the acoustic aspects of the sound diffusion and listening of the composed musical piece in a real concert situation. It is affected by technical criteria such as number and quality of available speakers, sound spatialization techniques and algorithms and acoustic qualities of the concert room. Nevertheless, the sound diffusion can also present an interpretative aspect. A musical piece that is conceived without a direct dependence of a specific location, as opposed to site-specific art works (Kaye, 2013), offers the possibility of being spatially interpreted. The sound diffusion can be controlled during the performance in order to give prominence to spatial aspects that could only emerge from the relationship between the composition and the performance location.

The balance between the pre-conceived spatial attributes of the musical composition and the interpretative potential that arises from the communion between the composition and the performance space is explained by Flo Menezes in what he entitles *the two laws of spatialization* (Menezes, 2006):

First law of spatialization:

> The bigger the number of loudspeakers in the concert room, moreover, the bigger is the difference between the number of available loudspeakers in the concert room and the number of channels that the composition was conceived for, the bigger will be the interpretative potential of the music in the concert situation, and consequently, the more relevant is the role of the sound diffusion performer.

Second law of spatialization:

> The interpretative potential is inversely proportional to the depth and complexity of the spatial attributes that are pre-defined in the musical composition. The bigger the precision and detail, both technically and aesthetically, of the spatial attributes in the composition, the smaller is the role of the sound diffusion performer in a concert situation.

A typical spatialization process will thus have two main stages. A creative stage, where the spatial ideas and concepts are elaborated intrinsically in the composition process, and an

implementation stage, where a spatialization algorithm is applied to the already developed and implemented audio processes in order to generate (render) a multichannel audio output.

The implementation process consists of feeding pre-designed trajectories (usually geometrical patterns such as spirals, lissajou curves, spherical movement, etc.) and an audio input source into an algorithm that calculates the amplitude of the input audio source for each of the desired output channels. Additionally, other audio processes such as doppler effect, reverberation and filtering can be part of the algorithm. The most common algorithms, that are adaptable for any concert situation are: Distance Based Amplitude Panning (DBAP), Vector Based Amplitude Panning (VBAP) and Ambisonics (Fellgett, 1975; Kostadinov et al., 2010; Malham, 1998; Penha, 2014; Pulkki, 1997).

This strategy forces a clear break between the development of electronic sounds and the development of spatial attributes. The spatialization algorithm lies completely outside of the audio processes both technically and also conceptually. The spatial trajectories are applied to an already shaped and polished sound. More importantly, the spatial trajectory doesn't have an specific relationship with individual audio processes. All processes are modified the same way by the spatialization algorithm, hence, there is no relationship between the spatialization attributes and the specific attributes of an audio process.

Using particle system to implement the live-electronics system offers the possibility of creating a conceptual relationship between spatial trajectories and specific attributes of audio processes. If the user designs the environment of the particle system as a representation of a concert room, the dynamic behavior of the particles can have a direct relationship with their position on the actual performance space. As a consequence, mapping the generic attributes of a particle to the specific attributes (as explained in section 4.3.2) creates a bond between the specific attributes of an audio process and its spatial trajectory.

This reveals a new horizon of possibilities for designing the sound spatialization. It affords the possibility for designing not only geometrical spatial trajectories, but also creating a spatial aesthetic topography that is unique for each audio process.

Another possible spatialization strategy that emerges is the possibility of dismembering an audio process into several smaller pieces (breaking a particle into other simpler particles)

and assigning each piece to an independent spatial trajectory. This is rather useful for audio processes that are implemented by simple digital signal operations that perform in parallel, such as reverb (Fonseca, 2013), and several synthesis techniques.

While providing new possibilities for handling both the conceptual and technical aspects of spatialization, a live-electronics implementation that uses particle systems may also adopt the traditional spatialization strategies. In which case the particle system implementation is also efficient due to the fact that it gives an intuitive interface for describing spatial dynamics. Ultimately, in a typical particle system implementation a complex dynamic behavior is always represented by geometrical shapes and patterns on screen.

Chapter 5

Case Studies

During the course of this research, the particle systems modelling method for live-electronics systems has been used in the development of the live-electronics of several mixed music compositions. The method not only evolved from the technical and aesthetic problems that arose from the practical work, but also evolved along side their development. The practical work was indeed the laboratory for testing, evaluating and evolving the method from an initial conceptual idea to the formalized state presented in chapter 4.

In this chapter, three case studies of live-electronics systems that used the particle system modelling technique are presented. Each case study discusses the live-electronics of a different composition by a distinct composer. The live-electronics for these compositions were developed in a professional collaboration between the musical assistant André Perrotta and the composers. The development of these works in a professional manner dictates that the results must not only serve for the personal academic and research objectives of the musical assistant, but more importantly, the final live-electronics had to meet high professional standards and respect the conceptual and aesthetic objectives of each composition.

Each case study is presented with a brief introduction of the composition, the conceptual briefing provided by the composer to the musical assistant, the particle-based modelling process and the respective technical implementation strategy. Also, a discussion of the final results and advantages and disadvantages of the utilized methodology is presented.

In order to assess the final outcome and overall achievement of the developed live-

electronics systems, the composers were asked to give a feedback regarding their perception of the overall development process and satisfaction with the final result. For that purpose they were asked the following questions:

- Did the developed live-electronics systems meet the composers conceptual and aesthetic expectations ?

- Did the development and implementation strategy affect or interfere with the instrumental composition process ?

The answer of the composers to these questions are not in any ways absolute or determinant of the overall success and validity of the proposed method, specially if taken into consideration that in a professional scenario, the final live-electronics had to fulfill their expectations regardless of the adopted development strategy. Nevertheless, their answers serve as indication of the overall satisfaction of the proposed method.

5.1 *O Farfalhar das Folhas*

O Farfalhar das Folhas by the composer Flo Menezes is a mixed music work for 1 flutist (flute in C, in G and piccolo), 1 clarinetist (clarinet in Bb, in Eb and bass clarinet in Bb), violin, violoncello, piano and live-electronics. It was commissioned by Miso Music Portugal and was first performed on July 3, 2010, at Teatro Nacional São Luiz in Lisbon, Portugal, by the Sond'Arte Electric Ensemble conducted by Jean-Sébastien Béreau, with Flo Menezes, Paula Azguime and André Perrotta controlling the live-electronics.

The development of the live-electronics system of *O Farfalhar das Folhas* was the starting point of this doctoral research. Its implementation, that uses particle systems as a modelling technique, served as the basis for developing the formalization presented in chapter 4.

5.1.1 About the Composition

O Farfalhar das Folhas is intended as an homage to the memory of the composer's brother, the poet Philadelpho Menezes. One of his most inventive poems is inserted in the border of a *catalogue* of poems, and, being read in the counter-sense of reading, reveals itself as something uncertainly inserted amid the leaves, as a kind of imponderable intromission inside the printed *catalogue*. By manipulating this poem – classified by him as an *intersemiotic* one, crossing its visual, verbal and sonic aspects –, we realize the image of an insect that wings against a glass window. The sound of a /r/ emerges as an uncertain inserted phoneme in the middle of the word "insect" (*inseto* = *insect*; *inseRto* = *inserted*), while one discovers the poems verse: "An insert moves swiftly against the laws of writing".

The composition by Flo Menezes is not in any way a musical version of the poem. Instead, the intention is to construct a kind of intersemiotic intersection with it. *O Farfalhar das Folhas* (*The Rustling of Leaves*), a *sine littera* work in which the poem, projected on a transparent leaf, cohabits the same space of the musicians without being literally "intoned" by the piece, deals with three human conditions, continuously moving, even if not always in a linear way, towards the last one – the greatest of all human desires: *constraints*, the *libertarian* act, and finally the aimed *freedom*. All those human conditions of living are structurally exposed in the main profile of the piece, which is based on one of the harmonic techniques of the composer, namely on his *cyclic modules* Menezes (1997).

To these three human states – *restriction*, *liberation*, and *freedom* – the piece associates respectively micro-articulated textures, extended durations, and finally resonances with their loosing itineraries, to which correspond three spectral sound treatments in real time: distorted shuffling with ring-modulation; time-stretching; and synthesis in real time controlled by the musicians themselves.

5.1.2 Conceptual Briefing

The development of the live-electronics for *O Farfalhar das Folhas* started after the composer had already created the main conceptual and musical trajectories that he intended to

translate into actual written instrumental parts and electronic sounds. Thus, the composer's briefing to the musical assistant presented very well defined conceptual and aesthetic objectives.

The live-electronics of the piece should account for three different types of audio processes: a real-time shuffling effect that would mimic the dynamic behavior of an insect striking his wings against confining walls, with an additional layer of ring-modulation and distortion for achieving a rough and turbulent aesthetic; a time-stretching effect that could exacerbate the duration of specific notes played by the instruments; a synthetic resonance that would enhance the overall harmonic profiles of the piece (the results of the composer's cyclic modules technique). Additionally, the synthetic resonance should offer the possibility of highlighting specific notes played by the instruments.

While shuffling, ring-modulation, distortion and time-stretching are standard and usual audio processing techniques (as discussed in section 2.3.5), the proposed synthetic resonance effect had to be developed from scratch in order to fulfill the composer's conceptual and aesthetic objectives.

In the composer's point of view the synthetic resonance should have a relationship with the main concepts that underlie the music and it should blend organically to the sonic aesthetic of the acoustic instruments. The resonance should be continuously dynamic, following the harmonic paths without perceivable breaks. The timbre (aesthetic) of the resonating sound should also be dynamic, giving the sensation that despite the fact that the harmonic content is the same for a determined period of time, the timbre and the overall dynamics are in constant motion. Moreover, the movement and dynamics of the resonating sound should also be influenced by the instruments.

The piece is divided into thirteen parts, entitled by the author as *Situations*. Each *Situation* reflects a different harmonic content resulted from iterations of the composer's cyclic modules technique (Menezes, 1997) applied to the main harmonic profile. Also, in each *Situation*, one of the instruments performs a solo that highlights the most important notes (frequencies) of the harmonic content of that specific part. The synthetic resonance should react to these specific notes played by the solo instruments, and in some way detach them

from the overall electronic sonic mass emphasizing their conceptual importance and acoustic perception, providing a sense of harmonic convergence towards that specific pitch.

5.1.3 Development

The development of the live-electronics system for *O Farfalhar das Folhas* was branched into distinct parts: development and implementation of the resonant synthesis; implementation of the distorted ring-modulated shuffling effect; implementation of the time-stretching effect; implementation of the overall system with a graphical user interface for controlling the overall progress of events.

Given that the composer's briefing for the resonant synthesis accounted only for its conceptual and behavioral aspects and the fact that there was no *a priori* elected choice of audio process that needed to be applied in its implementation, the development of this effect would benefit from a strategy that could model and implement the proposed dynamic behavior separately from the actual audio process(es) in charge of generating the respective sound output.

For that reason, the adopted strategy was to use particle systems to create a model of the resonant synthesis behavior. The model would serve as the medium for better understanding, experimenting and also importantly visualizing the different behaviors and reactions that the resonant synthesis had to perform.

5.1.3.1 Particle-based Model

The first step towards the construction of the particle-based model that represented the resonant synthesis behavior was the identification of all the distinct sonority states and sonority state transitions required for implementing the resonant synthesis.

Three distinct sonority states were identified:

Sonority State 1: The resonant synthesis produces a pitched complex sound of which the spectral content peaks at the specific frequencies of the respective harmony played by the acoustic instruments on that part of the composition.

The sonic output is harmonically static but the timbre and overall amplitude is in continuous motion. The motion is smooth and there is no perceivable break or discrete steps in the overall amplitude, timbre and harmony.

Sonority State 2: Specific notes played by the instruments, considered by the composer as paramount frequencies of the overall harmonic content, causes a reaction in the harmonic synthesis. This reaction manifests as a new sonority state of the resonant synthesis.

In this state, the resonant synthesis produces a pitched resolved complex sound of which the spectral content presents almost exclusively (or predominantly) the specific frequency that triggered the reaction. The sonority aesthetic is as similar as possible to that of the previous state.

Sonority State 3: The resonant synthesis is aesthetically and conceptually identical to the initial state, however, the pitch resolution effect present in the second state leaves an impression in the harmonic spectrum, which now presents a subtle inclination towards the frequency that was previously strongly resolved.

From the identification of the sonority states it was possible to imagine a particle-based model motivated by the idea that an ever-changing superposition of frequencies can be modeled from the behavior of a swarm of bees flying around a point of interest. Each bee has an individual and independent motion trajectory and behavior that together with all the other bees form a cohesive overall motion and behavior. Furthermore, a swarm of bees presents qualities such as distribution of tasks, collective and individual reaction and hierarchy (Karaboga and Akay, 2009), that would also help on the design of the resonant synthesis effect.

Each particle (or bee) is assigned with a harmonic pertinent note, chosen by a probability distribution function. For each part of the piece the universe of possible notes would change accordingly. The particles keep their birth note until an interaction event happens. The

interaction events are: the soloist instrument (different parts of the music have different soloists) plays a highlight-intended note (we detect the note by means of pitch tracking); the digital performer interacts manually on the system and inputs the highlight-intended note.

The ever-changing timbre sensation effect was achieved by creating a relationship between the motion of the particles, the interaction with the musicians, the volume and spatialization of each particles audio generating process counterpart. Our particle system is implemented in two dimensions, the particles perform a standard circular motion around the center of the system with a randomized radius limited so that they are usually far from the center. They also have random angular velocity and can move freely in the clockwise direction. The amplitude attribute of each particle is mapped to be inversely proportional to the radius of the circular movement. In the center of the system (radius = 0) the amplitude is maximum. Additionally the amplitude attribute of each particle is mapped to the geometric distance of the corners of the system, hence each particle is independently spatialized.

The choice of circular movement as opposed to the more realistic flocking motion of an actual bee swarm is justified by both aesthetic and optimization factors. In a generic concert room, with a limited sound spatialization hardware the precision with which a sound trajectory can be perceived is limited, hence the difference between the realistic model of the bee movement and a simplification using circular motion is not evident. The main idea of providing the sensation of constant and frenetic movement is still present in the circular motion with the bonus of sparing valuable computational resources that would otherwise be wasted if the realistic mathematical model was used.

When an interaction event occurs, particles that hold the highlight-intended note (as described in sonority state 2) start a different movement. The idea is to use this motion to highlight and detach the specific note from all others creating a sense of pitch resolution in the resonance. This is achieved by forcing the particle to move towards the center of the system by operating on the radius parameter. When the particles reach the center, they stay there for a brief instance of time and then return to their usual peripheral movement. Due to the fact that the amplitude attribute reaches the maximum value in the center, during the period of time that the particle is moving to the center, staying there and moving back, its

amplitude is a lot bigger than the amplitude of the peripheral particles, hence achieving the desired effect.

At the end of this detachment movement, when the particle reaches the usual peripheral trajectory, the frequency attribute of all other particles that did not participate on the movement are reset to a new value using the probability function. Redistributing the frequency values creates the sensation of an ever-changing harmony that in fact always hold the same interval relationships, as determined by the composer. To avoid glissandos and breaks on the amplitude and harmony, each particle performs an independent short term (100ms to 500ms) fade-out and fade-in in amplitude while changing its frequency.

By keeping the particles that performed the detachment movement with the same frequency, the overall spectrum after all other particles receive new frequencies, presents stronger amplitude peaks at those specific detachment frequencies. In other words, the number of particles holding the detachment pertinent frequency in sonority state 3 is always bigger (or with a lot smaller probability, equal) than in sonority state 1. Thus the pitch resolution achieved in sonority state 2 remains (though fragmented) in sonority state 3.

This effect is radically explored in the last section of the composition. In this section the instrumental writing mainly consists of a cello solo and the electronic is generated by the resonant synthesis. The solo orbits around the *B* note and encourages the soloist to improvise towards the complete collapse or towards unison of the *B* note. By using the *B* tuned to a 493.88Hz frequency as the detachment frequency over and over again, the resonant synthesis also migrates towards the unison. Given that each time the systems goes through the loop of sonority sonority state 2, followed by sonority state 3, and back to sonority state 2, and so on, the number of particles holding the *B* note becomes bigger at each iteration until finally all particles are set with this same note.

The soloist is instructed to be aware of this effect and how he can in fact control the time that the system would take to achieve unison by avoiding or precisely using the *B* note in his improvisation. Additionally, the soloist is instructed to stop playing as soon as he/she perceives that the resonant synthesis has reached the unison state. This marks the end of the instrumental performance and the actual overall musical end is a slow manual fade-out of

the resonant synthesis sound.

Figure 5.1 shows the graphical visualization of the particles (bees) performing a complete cycle of sonority states 1, 2, and 3.

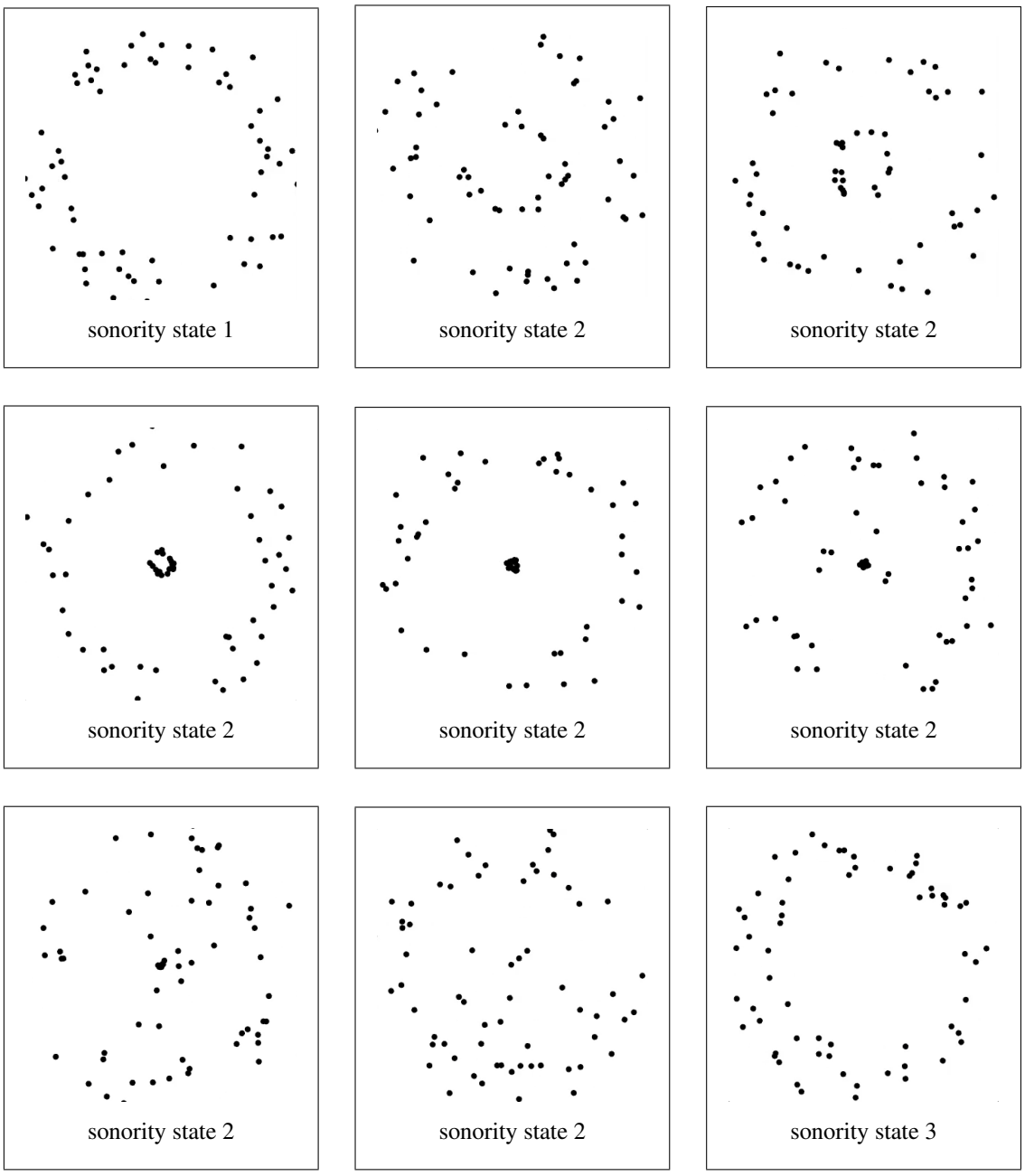


Fig. 5.1: Graphical representation of particles in sonority states 1, 2 and 3 of the resonant synthesis of *O Farfalhar das Folhas*. The picture represents a top view of the 2D particle system environment.

5.1.3.2 Software Implementation

The live-electronics system for *O Farfalhar das Folhas* was developed using the Max programming environment and the particle system implementation of the resonant synthesis was developed in C++. The reasons for choosing this configuration of software development languages and environments has already been discussed in section 4.3.3.

The software was designed to provide simple and straightforward calibration and performance routines. The synchronization between electronic and instrumental parts were developed using a manual triggering global control structure (see section 2.3.3). Each global event is clearly indicated on the score and can be directly triggered via the software GUI, which in turn presents an individual button for each event. Allowing the digital performer to go freely back and forward on the events, which in turn facilitates the use of the electronic parts during rehearsals. Additionally it allows for quick correction of synchronization during the performance, in case mistakes are made either by the instrumental ensemble or the digital performer.

Figure 5.2 displays the graphical user interface of the live-electronics software for *O Farfalhar das Folhas*. The left side of the GUI contains the individual trigger buttons for all global events and the buttons for accessing other relevant features such as audio on/off, access to the processing patches, among others. On the right side the 2D audio spatialization graphical representation and overall audio output levels are displayed.

Along with the adoption of manual triggering for the global control structures, all local control structures were developed to be completely automated, therefore the digital performer did not need to control any variable parameters during the performance. As discussed in section 2.3.4 this strategy has the characteristic of freeing the digital performer of any level of artistic interpretation, hence the electronic part can be executed by any technical operator capable of following the instrumental score.

The particle-based model for the resonant synthesis is implemented in C++ as a standalone application and its audio processing counterpart was developed using Max visual programming language. The communication between both parts is done with the OSC protocol. The audio process that represents each particle is implemented using a pair of sine

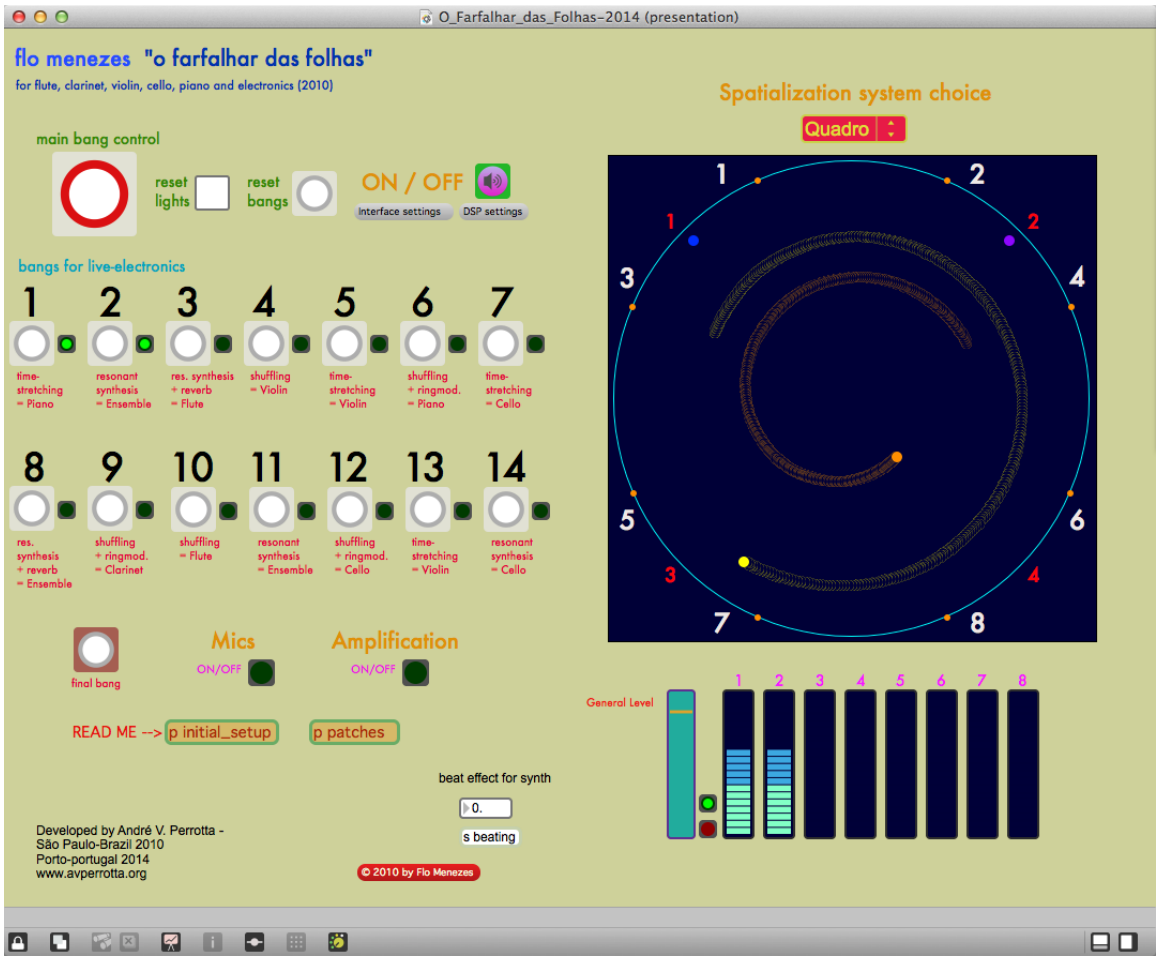


Fig. 5.2: Graphical user interface of the live-electronics software for *O Farfalhar das Folhas* displaying the individual trigger buttons for all global events, the 2D audio spatialization graphical representation and overall audio output levels.

wave oscillators fed into an independent DBAP audio spatialization engine (see section 4.3.4.5). The audio process is encapsulated using the **poly**~ object and both the particle system and the **poly**~ object are initiated with 64 particles and instances respectively.

Each particle controls a specific instance of the the designed audio generating process and therefore the particles are configured with the generic and specific attributes (as discussed in section 3.3) shown in table 5.1.

The use of a secondary oscillator in addition to the one responsible for generating the harmonic pertinent tone (note frequencies that are chosen by a stochastic distribution relative to the harmony of each part of the piece), is justified by aesthetic reasons. A second

Particle type	Generic attributes	Specific attributes
resonant synthesis particle	index	harmonic pertinent note frequency
	status	harmonic pertinent note amplitude
	position 2D	frequency disparity
	velocity 2D	secondary oscillator amplitude.

Table 5.1: Generic and specific attributes for *O Farfalhar das Folhas* resonant synthesis particles

oscillator with very small frequency disparity to the main oscillator frequency results in periodic amplitude modulations (beats) and also changes in the timbre and pitch stability. These modulations are proportional to the difference between frequencies and the way that this difference changes over time (Menezes, 2004). The aesthetic consequence is the avoidance of the typical homogeneity and predictability of the computer additive synthesis timbre. The acoustic beats derived from the interference between oscillators generate a more natural timbre and tone sensation that in turn is easier to blend with the sound generated by the acoustic instruments.

Figure 5.3 displays the signal processing chain for the audio generating process that is controlled by each particle and generates the overall resonant synthesis used in *O Farfalhar das Folhas*.

The detection of the interaction event was implemented using Miller Puckette's **fiddle~** object (Puckette, 1997), which performs reliable pitch tracking for monophonic sounds. The continuous detected pitch is fed into a gate object that is configured to pass specific frequencies. Every time the gate passes on a frequency, a trigger message is delivered to the particle system and all particles that have the trigger frequency set as their harmonic pertinent note value, change their status attribute and start describing the behavior described as sonority state 2 (section 5.1.3.1).

5.1.4 Discussion and Composer Feedback

In the development of *O Farfalhar das Folhas*, the use of particle-based modelling for achieving some of the proposed sonic concepts and aesthetics, namely the resonant syn-

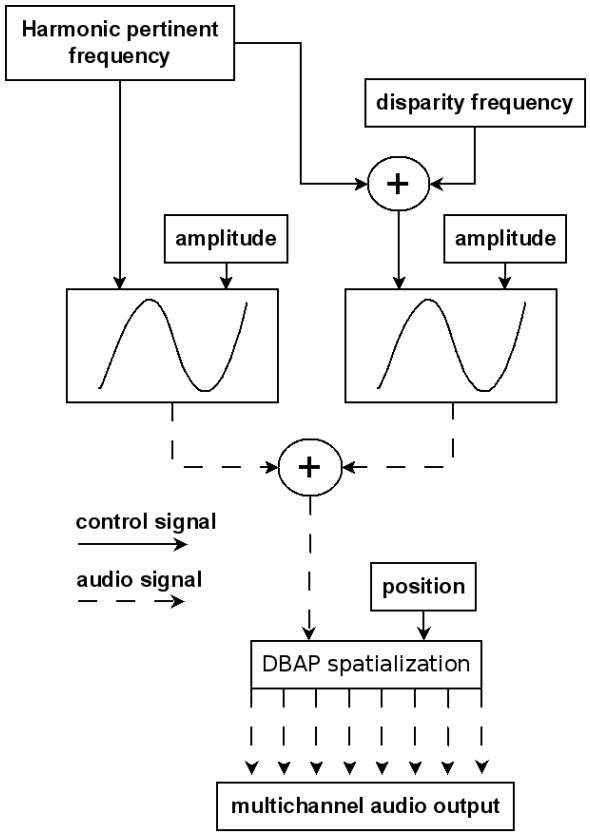


Fig. 5.3: Audio processing diagram for the *O Farfalhar das Folhas* resonant synthesis particle.

thesis effect, was determinant.

The conceptual briefing proposed by the composer presented a complex web of interlaced sonic aesthetics, intersemiotic concepts and musical structures. The particle system modelling strategy provided a framework for de-interlacing this web and creating an implementable structure for each of the decisive underlying attributes, such as the cohesive harmonic continuity, the pitch resolution, the diffused spatialization, among others.

From a more assertive research perspective, the use of particle systems in this piece served as a testing ground for evaluating the previously discussed stochastic transitions and spatialization features of the method (as presented in sections 4.3.4.5 and 4.3.4.4). The former is present in the transitions between sonority states (section 5.1.3.1) and the later is present in the actual resonant synthesis audio process implementation, where each instance of the implemented synthesis processing chain is provided with an individual spatialization module.

In both cases, these intrinsic features of the particle system modelling method bolstered the achievement of the complex dynamic behavior of the harmonic content, providing the framework for implementing the diffused ever-changing and continuous motion of the harmonically static content and its transition into and out from a clear pitch resolved state.

From the composer point of view, the final result of the live-electronics satisfied the conceptual and aesthetic criteria aspired for the electronic part of the composition. He also states that the use of new methods and strategies for implementing the live-electronics opened up new possibilities that were not obvious (or evident) *a priori*.

The new possibilities that emerged from the design of the particle-based model also impacted the instrumental writing. Specifically in the final part of the composition, where the possibility of having the resonant synthesis effect gradually converging to unison, by interacting with the solo instrument, encouraged the composer to rewrite the instrumental part to take advantage of this possibility.

On this matter, it is the composer's opinion that composers should always embrace and take advantage of unforeseen technical solutions that are revealed during the composition and development processes. Moreover, the composer points out that the confrontation of composition and technology is present throughout the history of music. In his words:

> "The history of music is the history of the dialectic between composer and mediums, whether it is acoustic or electronic." (Flo Menezes, 2015)

5.2 *Changeless*

Changeless by the composer Paulo Ferreira Lopes is a mixed music work for Timpani and live-electronics. Dedicated to the percussionist Nuno Aroso, the piece is the result of extensive exploration of the timbral and sonic possibilities of the Timpani. This work was first performed on July 7, 2011 at Casa da Música in Porto, Portugal, by Nuno Aroso (percussion) and André Perrotta controlling the live-electronics.

5.2.1 About the Composition

Changeless tries to explore non traditional timbre and sonority possibilities of the Timpani instrument by using objects other than the traditional drum sticks such as bow and rubber balls to vibrate the instruments membrane and body. Additionally, different types of bells are used on top of the membrane to generate high pitched notes.

The composition is divided into two parts which can also be understood as two different compositions, *Changeless I* and *Changeless II*. Even though both uses the same timbral and sonority resources, each one emphasizes specific characteristics.

In *Changeless I* the instrumental writing is focused on the exploration of long and harmonic rich glissandi sounds, that are achieved by rubbing objects on the Timpani membrane and controlling the pitch with the instrument pedal. Furthermore, the writing is rather graphical and there is no precisely defined (or absolute) pitch definitions, the main priority is on the expressiveness of the glissandi and on the morphology of the sounds rather than their harmonic precision. Figure 5.4 shows the first page of *Changeless I*.

In *Changeless II* the composer explores the expressiveness of noisy and unpitched breath-like sounds, which the player extracts by using a bow on the Timpani metal structures and body. The composer uses this effect to create an organic relationship between the player respiration and the instruments vibration. Figure 5.5 shows the beginning of *Changeless II*, where the composer merges the performer and the music respiration by the use of periodic repetitions of the described sound effect.

In both pieces the composer uses sparse and punctual percussive attacks on high pitched bells in order to create contrast and therefore highlight the thick and low sounds of the Timpani and the unpitched noisy breath-like sounds.

5.2.2 Conceptual Briefing

The live-electronics of *Changeless* (I and II) was developed based on the concept that the electronic sounds should not directly interfere or transform the acoustic instrumental sounds, rather it should serve merely as an echo or "distant memory" of some instrumental events.

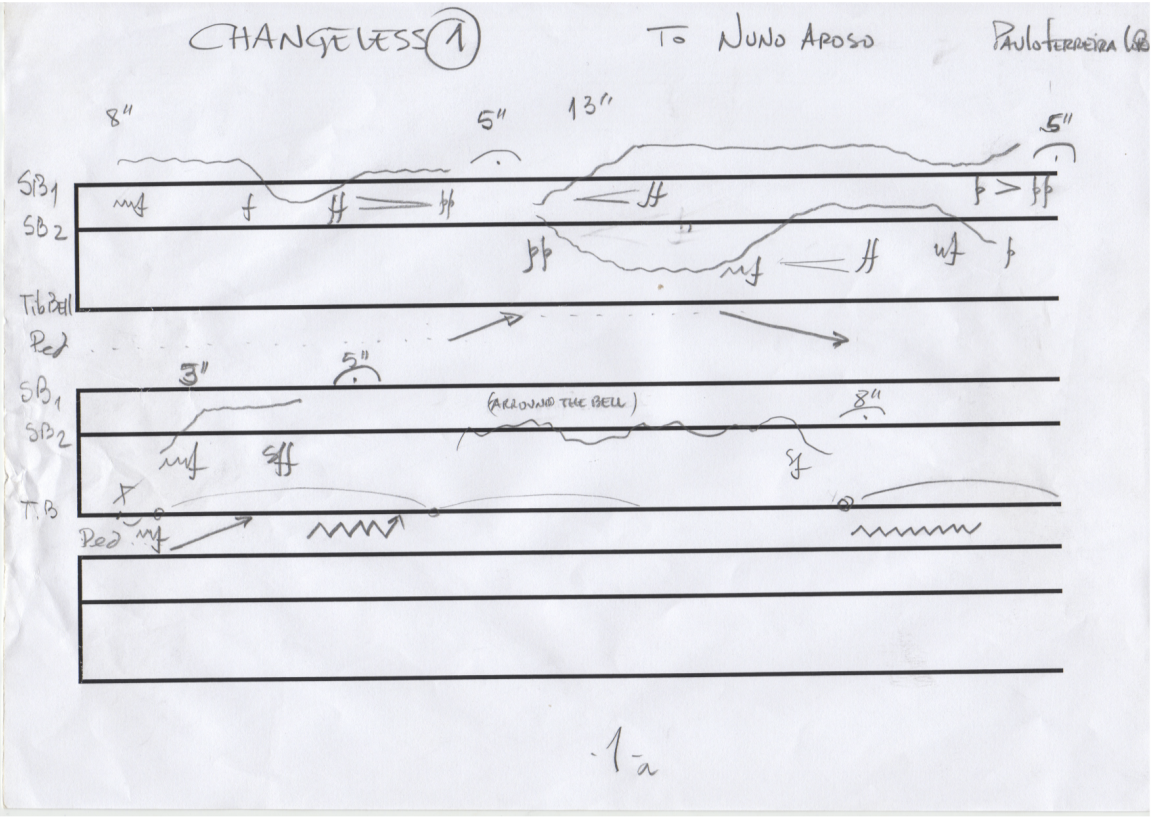


Fig. 5.4: First page of *Changeless I* by Paulo Ferreira Lopes. The writing is mostly graphical and focused on expressiveness

Additionally, the electronic sounds should be able to fill the moments of instrumental silence, which were a consequence of the abundant use of fermatas[1], acting as the connection between sonic events that are separated in time. The overall flow (tempo) of the instrumental part is freely controlled by the percussionist, hence the live-electronic could not be fixed to pre-determined events, on the contrary, it should be able to freely follow the percussionist.

Also important, it should be susceptible to the artistic interpretation of the digital performer.

The aesthetic qualities of the electronic sounds should allow for an organic blend with the instrumental sounds so that it could be used as the continuation of the sound of the acoustic instrument, the enunciation of an imminent instrumental sound, or as a background layer with complete independent behavior, but with no conflicting harmony or timbre.

[1]Fermata is the traditional music notation symbol for indicating that a note or rest should be freely prolonged by the performer or conductor (Kennedy, 1994).

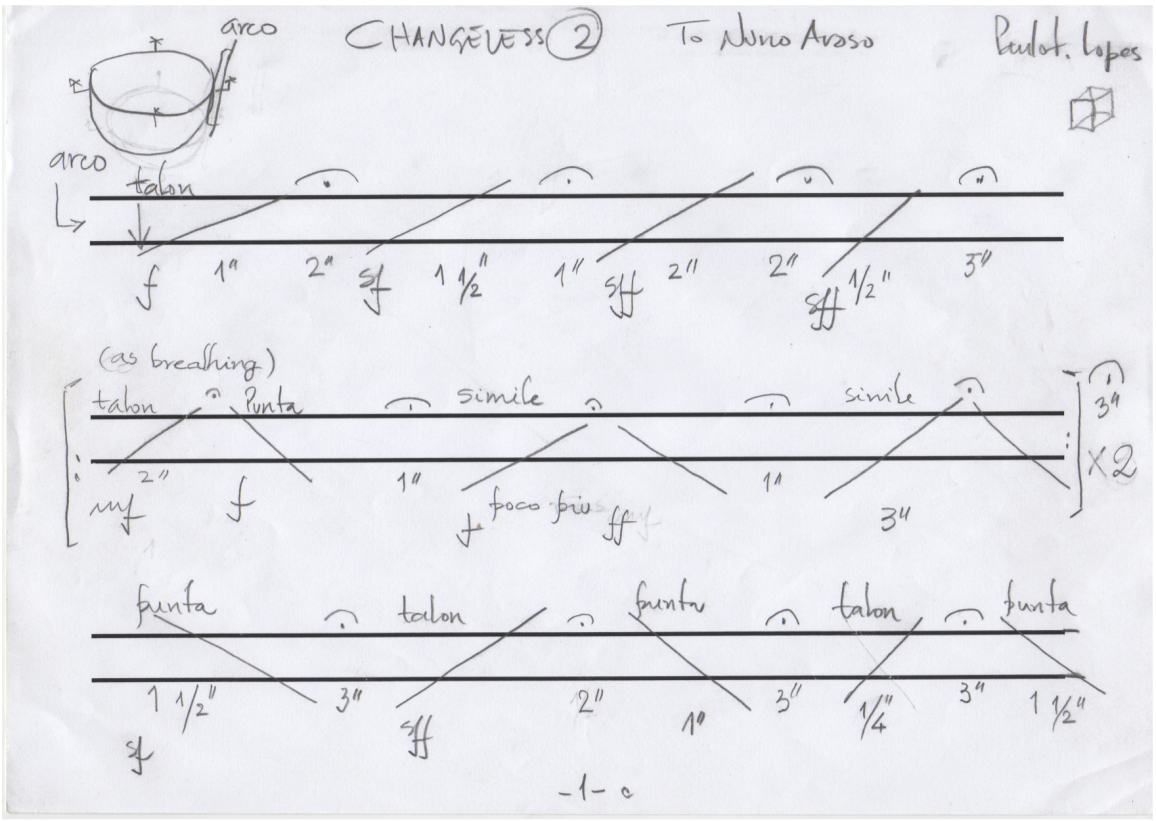


Fig. 5.5: First page of *Changeless II* by Paulo Ferreira Lopes. Repetitions of the same morphology applied to a noisy and unpitched breath-like sound and the use of *fermatas*, proposes the symbiosis between the performer respiration, the musical mood and pulse and the instrument vibration.

Another important aesthetic aspect was that all electronic sounds should have very smooth attack and decay (relatively long fade-in and fade-out), and they should be spatialized in the concert room in a diffused manner. The diffuse spatialization and smooth and continuous transitions of presence and absence of electronic sounds should create a clear contrast with the concrete physical presence of the percussionist and instruments, which in turn delineated a precisely fixed and localized sound source.

5.2.3 Development

The composer's conceptual briefing for the live-electronics did not specify any audio process or control form that should be implemented. Rather, the briefing was focused on the overall aesthetic and behavioral qualities of the final electronic sounds, rather than their precise

description and localization in time. Hence the main challenge on the development for the electronic sounds was to create audio processes that could be seamlessly controlled by a digital performer, providing access to complex morphological behaviors through very simple controls.

In order to achieve the depicted objective, the adopted strategy was to use particle systems to create a model that could perform the spatialization, diffusion and behaviors as briefed by the composer. The model would serve as the guiding path for creating the audio processes that would best fit the composer's aesthetic intentions as well as serving as a mapping layer between simple user input controls and the complex dynamics of multiple audio processes parameters.

5.2.3.1 Particle-based Model

From the composer's briefing and the conceptual aspects of *Changeless I* and *Changeless II* it was possible to identify the unique sonority states that needed to be implemented.

Sonority State 1: The electronic part produces a spatially diffuse harmonically rich sound that recounts previous instrumental sounds in terms of harmony and timbre. These electronic sounds are triggered by a digital performer that freely chooses their placement either as a reaction or as an anticipation to the acoustic instrumental sounds produced by the percussionist when rubbing the Timpani with rubber balls or attacking the high pitched bells placed on top of the Timpani membrane.

Additionally, the digital performer should be able to trigger multiple layers of this electronic sound, each with its own dynamic envelope providing different duration, amplitude, attack and decay times.

Sonority State 2: The electronic sounds produce a spatially diffuse dilatation of the breath-like noisy acoustic sounds produced by the performer by applying a bow to the Timpani metal structures. The aesthetics of these electronic sounds amplify the rough and grasp qualities of the acoustic sound but at the

same time present a clear contrast that allows for the listener to clearly identify the different nature of both sources (acoustic and electronic).

Similarly to sonority state 1, the digital performer should be able to trigger multiple layers of this electronic sound, each with its own dynamic envelope providing different duration, amplitude, attack and decay times.

From the identified sonority states it was possible to observe that both sonority states present similar behavior characteristics, and their distinction is intrinsically aesthetical. Therefore both states could be described by the same model.

The particle-based model was formulated supported by the idea that the described sonic behaviors resemble a smoke emitter in an almost completely isotropic environment, where the emitted particles perform a smooth, long lasting and long fading random trajectory starting from a focused localized emitting source.

Given that the objective was to develop a model for audio generating purposes as opposed to realistic applications or graphical rendering, the smoke particles emitter model could be significantly simpler than a physically realistic model constructed with fluid dynamics equations (Stam, 2003). The simplified model only needed to account for emitting particles that have random time-to-live values and random diffused movement trajectories.

In order to implement the necessary audio processes that perform the sounds described in sonority states 1 and 2 using the same particle system, it was necessary to create different types of particles with identical generic attributes. Hence, while their behavior is similar, their specific attributes and the mapping between the specific and generic attributes would allow for generating aesthetically different sounds.

The harmonic rich sounds described in sonority state 1 suggested that this state should be implemented using additive synthesis. This could be solved (similarly to the implementation of the resonant synthesis used in *O Farfalhar das Folhas* (see section 5.1)) by assigning each particle with oscillators set to a specific frequency. However, the aesthetic of a simple additive synthesis, even if each component of the spectrum has independent dynamic envelope and spatialization, is notably smooth and uniform.

Due to the intrinsic acoustic characteristics of the Timpani instrument and the method which was used by the percussionist to generate the sounds (rubbing the membrane), the instrumental sounds that needed to be reciprocated in the electronic part presented a much more rough and even noisier quality, therefore a simple additive synthesis wouldn't suffice. To overcome this problem, the strategy was to use FM-synthesis (discussed in section 2.3.5) instances instead of simple sinusoidal oscillators.

Additionally, in order to generate synthetic sounds with the same (or very similar) harmonic spectrum of the acoustic input, the implemented audio process should be affected by the tracked spectrum of the live instrumental audio.

The dilatation of the noisy breath-like sounds described in sonority state 2 suggested that this state could be implemented with a time-stretching audio process. The live instrumental input of this sonority state was characterized by radically weak (barely audible) audio input that resembled a bandpass filtered pink noise. This subtle but rich spectrum sound would be easily masked by an overpowering electronic sound, hence, using the particles to superimpose several layers of time-stretching audio processes was not the elected choice.

Instead, the adopted strategy used a single instance of a time-stretching audio process that was spatially diffused by chopping its output in several frequency bands and assigning each band to an independent spatial trajectory and dynamic envelope, which in turn was controlled using the particle system.

As a result, the particle system should be implemented with two different types of particles presenting the following sets of specific and generic attributes, shown in table 5.2:

5.2.3.2 Software Implementation

The live-electronics system for Changeless was developed using the Max programming environment and the particle system was implemented as a stand-alone C++ application that runs on background and the communication between both applications is done with the OSC communication protocol.

The main Max application contains the audio processes implementation and a simple GUI that enables quick and straightforward setup and calibration. There is no global control

Particle type	Generic attributes	Specific attributes
FM-syntehsis particle	status time-to-live age position 2D	carrier frequency harmonicity amplitude
Time-stretching particle		center cutoff frequency bandwidth amplitude

Table 5.2: Generic and specific particle attributes of the particle system implementation used in *Changeless*

structure for controlling the synchronization between instrumental and electronic sounds, rather, the digital performer is able to trigger electronic sounds generated by synthesis or by time-stretching at any instant, using the computer keyboard or by clicking on the trigger buttons on the GUI (figure 5.6).

With respect to the synthesized sounds that implement sonority state 1, the digital performer can trigger events of five different durations (ranging from 6 to 45 seconds), all of which have a gaussian amplitude profile. Each time a new synthesis event is triggered, new FM-syntehsis particles are injected in the particle system and reciprocally the same number of FM-synthesis audio process instances are added to the audio processing system. As a result, the sound generated by successive events can be overlayered.

Additionally, every time a synthesis event is triggered, the system extracts the most significant partials form the spectrum at that instant. The injected FM-synthesis particles set their carrier frequency attribute by choosing randomly from a universe that contains the tracked partials and octave multiples (3 successive higher octaves and 2 successive lower octaves, $(f_0, f_0 \times 2, f_0 \times 3, f_0 \times 4, f_0 \times 0.5, f_0 \times 0.25)$). The digital performer also has the option to pre-multiply the tracked frequency components in order to better control the height of the generated sound.

The harmonicity attribute value, which is the most significant parameter for shaping the timbre of the overall synthesized sound, is set by a smooth randomic amplitude and frequency sinusoidal function mapped to the particles velocity and constantly generates

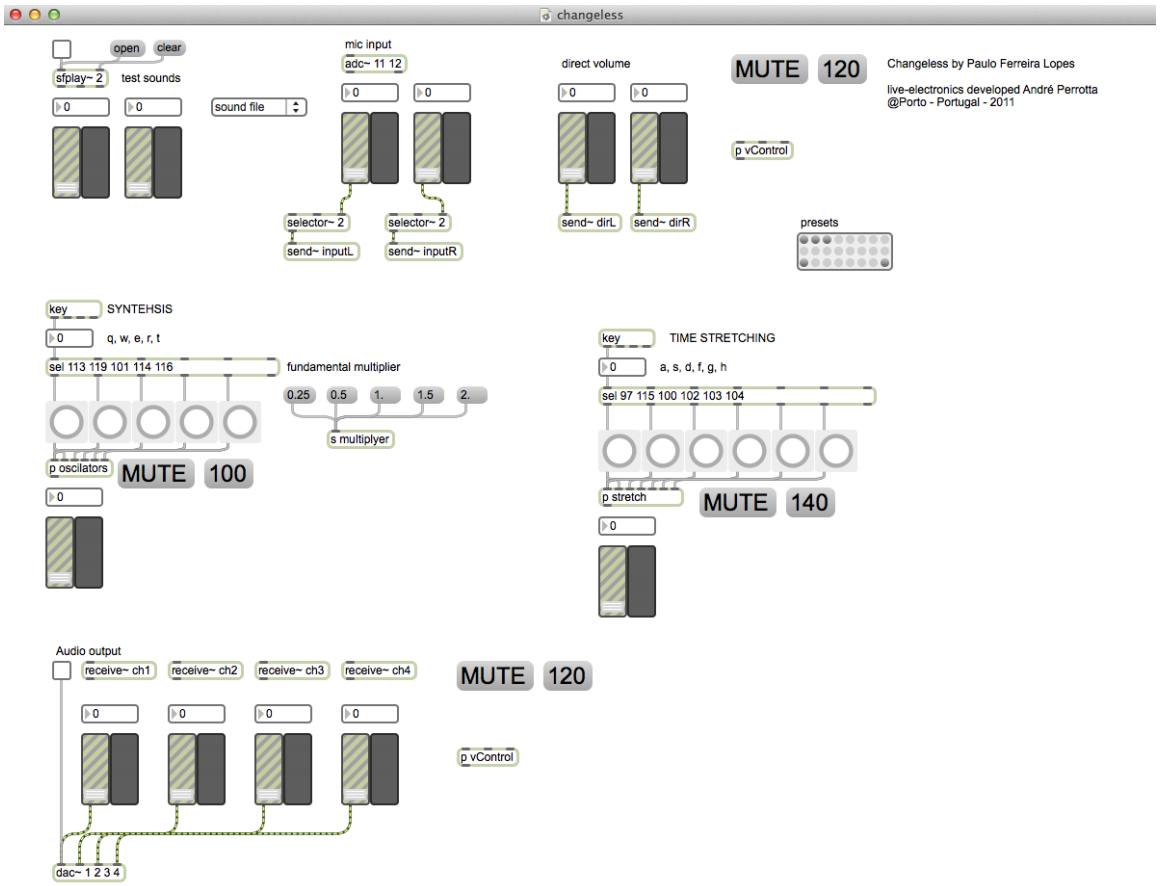


Fig. 5.6: Graphical user interface of the live-electronics software for *Changeless*.

(while the particle is still alive) low values (between 0.2 and 0.3, beeing 1.0 the value that transforms the FM-synthesis in a simple additive synthesis). This enables the system to have at the same time FM-synthesis instances with harmonic and inharmonic partials.

Figure 5.7 displays the audio processing chain of the FM-synthesis particle audio process.

Regarding the time-stretched sounds that implement sonority state 2, the digital performer is also provided with the possibility of triggering events with five different durations (ranging from 1 to 30 seconds), shaped with a gaussian amplitude envelope. However, as opposed to the synthesized events, overlapping is not allowed. Every time a new time-stretch event is triggered and there is still a previous time-streching event generating sound, the system creates a short crossfade between the new event and the previous one. The crossfade is designed with a profile that enables the perception of discontinuity of sound, rather than

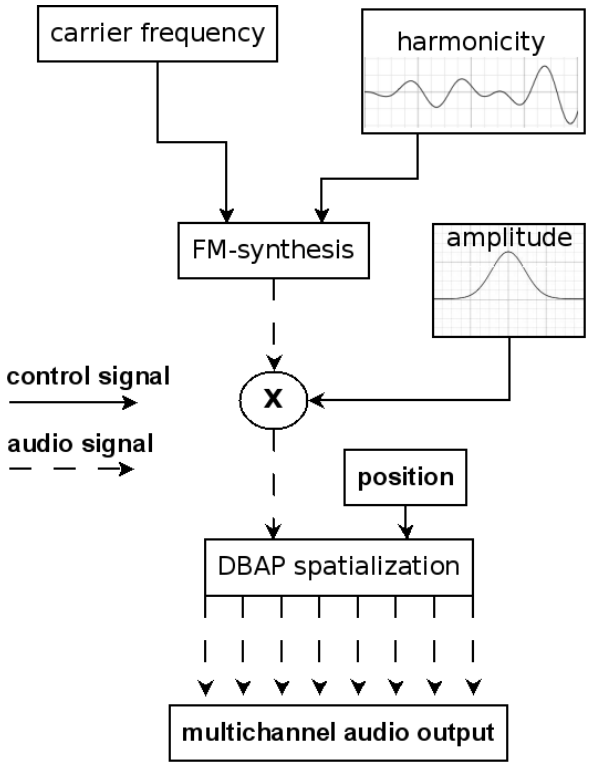


Fig. 5.7: Audio processing diagram of the FM-synthesis audio process used in *Changeless*

creating a smooth continuous transition. This particular behavior helps the digital performer achieve a more natural relationship with the rhythm of the instrumental breath-like sounds (previously described in section 5.2.1).

The time-stretching audio process is implemented using a standard granular synthesis algorithm (N. Schnell, 2005) and the relevant parameters such as grain size, buffer size and stretch factor are manually set by aesthetic criteria. The audio output of the granular time-stretching audio process is then fed into bandpass filters, that in turn are controlled by particles in the system. Thus each time-stretch particle is actually controlling one instance of a bandpass filter set with unique center frequency and bandwidth. The center frequency and bandwidth attributes of each particle is set using a distribution criteria so that the whole spectrum range (40Hz to 20kHz) is covered by the aggregation of particles. The output of each filter instance is then fed into an independent DBAP spatialization algorithm. Figure 5.8 displays the DSP diagram of the encapsulated audio process that is controlled by the time-stretching particles.

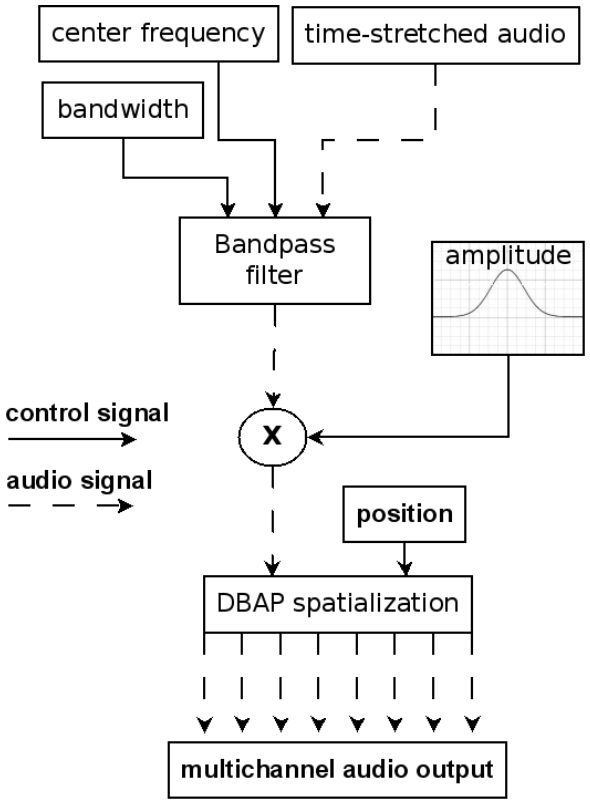


Fig. 5.8: Audio processing diagram of the time-stretching particle audio process used in *Changeless*

5.2.4 Discussion and Composer Feedback

The biggest challenge in the development of the live-electronics for *Changeless*, was the implementation of a control structure that enabled the digital performer to react or interact with the instrumental performer in an natural and seamless way. The digital performer needed to be able to create (trigger) complex and dynamic electronic sounds, that performed either in complete synchrony with the instrumental part or with total time independence.

The use of the particle system modelling method proved to be very effective for overcoming this challenge. Using the same system and also particle-based model for both the synthesis and time-stretching sonorities, implemented with a very simple event triggering strategy, enabled the digital performer to produce a wide scope of sonic interactions, ranging from very punctual rythmic to long lasting and multi-layered events.

This range of possibilities was a direct consequence of the intrinsic micromodulations and layering features of the particle system modelling method (as discussed in section

4.3.4.3). Achieving distinct sonic results with the same audio process, usually requires the direct control over a great number of variables (already discussed in sections 2.3). With this technique, the direct control over the audio process variables is substituted by the direct control of the life span and instant of injection of the particles. As a consequence it is possible to use simple interaction strategies for generating a wide range of sonic outputs. In the live-electronics of *Changeless*, this becomes evident in the simple triggering structure that is implemented for the synthesis and time-stretching events, where the digital performer need only to choose the duration of the sonic layer and the instant that it is triggered.

In the composer's perspective, the final result of the live-electronics perfectly matched the conceptual and aesthetic aspects aspired for both compositions. Furthermore, it is his assumption that the way the final software was implemented, providing the digital performer with the ability to freely interact with the instrumental performer through an interface that mapped very simple commands to a wide range of complex sonic aesthetics, enabled the discovery or enlightenment of conceptual and aesthetic relationships between electronic and acoustic sources, that were not initially planned.

The composer also highlights the fact that the technical development strategy did not impose any limits or constraints to the musical ideas, and also enabled a seamless concert performance.

5.3 *Nomoi*

Nomoi by the composer Tiago Gati is a mixed music work for solo violin and live-electronics. It was commissioned for the DVD Project *Boulez+* of the Studio PANaroma for the label SESC - SP, and it was first performed on November 13 and 14, 2013, at Teatro Sesc Anchieta in São Paulo, Brazil by Elissa Cassini (violin) and Tiago Gati and Flo Menezes controlling the live-electronics.

5.3.1 About the Composition

Nomoi expresses the composer's exploration of the limits and boundaries where the acoustic and electronic sounds interface. The composition tries to express a journey from a well defined, compartmented, normalized and ordered sonic situation - *nomos*, in the ancient Greek meaning of the word - to a complete fragmented sonic situation, where distinct sonic and expressive aspects of the violin are distributed in the concert space, providing the audience with a fragmented or multifaceted perception of the acoustic instrumental sound.

In order to achieve this goal, the composer uses intense repetitions of the same instrumental material, but with distinct electronic transformations that extracts, detaches and spatializes specific aspects of the instrumental sound or harmonic content. Thus amid the repetitiveness of the instrumental material, the electronic sound becomes the protagonist of the overall sound, and indeed breaks away from the fixed and constraining *nomos*.

Hence the instrumental writing of *Nomoi* is strongly characterized by the use of repetition of musical phrases and the constant presence of specific notes that are constantly prolonged and interlaced in the middle of articulated phrases, thus acting as a harmonic force layer, at times pushing towards consonance, and other times pushing towards dissonance.

For these repetitive phrases and long lasting pedal notes, distinct temporal distributions were used. Short articulated melodic phrases are repeated over time with none or subtle melodic variations. Between each repetition the composer uses either pauses of different durations (Figure 5.9), long lasting chords that are either static or modulated by glissandi or other types of spectral and dynamic modulations such as *crescendos*, *sforzandos*, *arco molto vibrato*, among others (Figure 5.10). Or by combining several repetitions in one long circular motion (Figure 5.11).

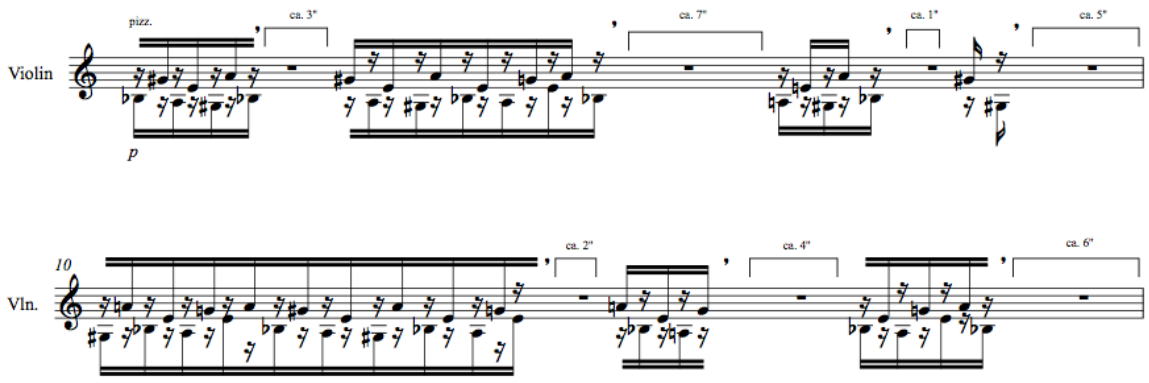


Fig. 5.9: Excerpt from *Nomoi* displaying repetitions of the same melodic structure separated by pauses.

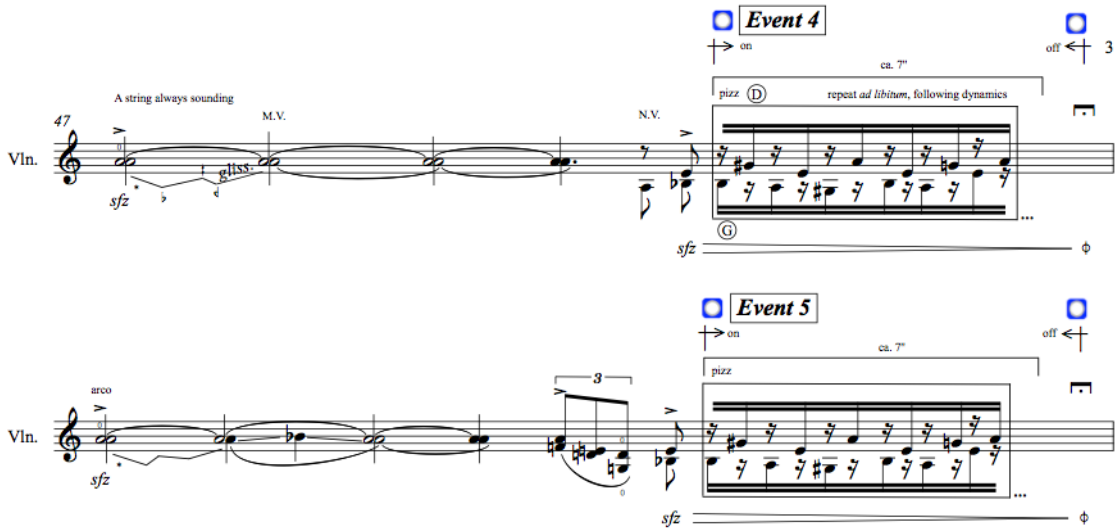


Fig. 5.10: Excerpt from *Nomoi* displaying repetitions of the same melodic structure separated by long lasting chords.

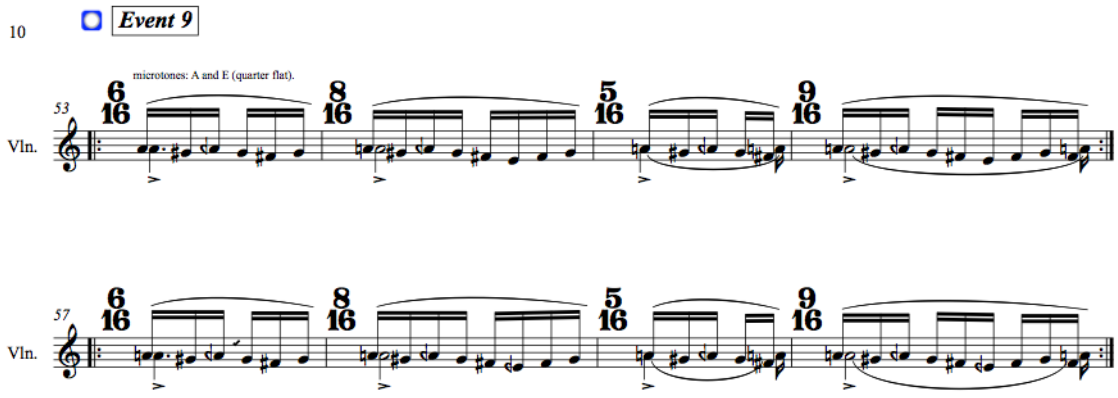


Fig. 5.11: Excerpt from *Nomoi* displaying cyclic repetitions of the same melodic structure.

5.3.2 Conceptual Briefing

The live-electronics of *Nomoi* was developed after the composer had already written most of the instrumental part, furthermore the composer had very specific ideas for the electronic transformations, how and where they should perform on the piece.

The composer specified two main electronic effects that he wanted to be present in the composition: a pizzicato effect, that would mimic and reference the already pizzicato instrumental sound that is present throughout the piece, thus providing the listener with a contrast between an acoustic effect produced by the instrumental player and an electronic simulation of the same musical gesture applied to other parts of the piece. The pizzicato effect should also transpose the incoming notes towards the lower register and create spatially distributed repetitions of each note. And an analysis-resynthesis resonance effect that could stretch each individual note of the articulated melodic phrases and long lasting chords in time and space, at times with consonant harmonic spectrum and at other times with a dissonant harmonic spectrum.

The analysis-resynthesis resonance effect as described by the composer should be able to accomplish several adverse conceptual tasks and also it should be able to adapt to distinct aesthetic aspirations. With that regard, it should serve as the connective element between repetitive structures that were separated by pauses, it should serve as a background "sonic shadow" and thus by contrast evidence the instrumental part, and it should also amplify or

highlight the embellishment and modulation of the long lasting chords.

Additionally, the composer wanted this synthetic sound to be able to "morph" between very smooth additive synthesis like sounds that clearly contrast with the instrumental timbre, passing by a resonant aesthetic that could (at least ideally) completely fuse with the instrumental timbre, and finally reaching very high pitched "microphone-feedback-like" sounds, that would "tear down" the harmonic fabric.

5.3.3 Development

The pizzicato effect described by the composer could be directly implemented using a combination of an attack detector, delay lines with feedback and modulated delay times, and a spatialization strategy analogous to that of a classic stereo-ping-pong delay effect. Thus its implementation was straight-forward and did not present a complex conceptual-mapping development.

On the other hand, in order to achieve the analysis-resynthesis resonance effect, the adopted strategy was to create a particle-based model that could describe the different conceptual and morphological trajectories envisaged for this effect and from that model develop and test different audio processing and synthesis strategies that could finally represent the composer's aesthetic aspirations.

5.3.3.1 Particle-based Model

The composer's briefing described the different roles that the analysis-resynthesis resonance effect should perform during the piece, also, it described the panorama of the possible sonic aesthetics that it should accomplish. From that description it was possible to identify the unique sonority states that needed to be implemented.

Sonority State 1: The electronic part stretches in time and space each note played by the violin, creating a smooth sounding harmonic resonance that is consonant to the instrumental phrases. Each stretched note should have very smooth fade-in and out amplitude envelope, giving no sense of precise

attack or decay.

The aesthetic characteristic of the sonic result is similar to a smooth additive synthesis, dissonant frequencies should be avoided and also the sound should be harmonically stable, with no periodic amplitude or frequency modulations such as glissandi, vibrato, tremolo, etc.

The frequency spectrum of the synthesized sound should encompass not only the fundamental detected frequency, but also harmonic partials higher and lower than the fundamental.

Sonority State 2: The electronic part stretches in time and space each note played by the violin, creating a harmonic resonance that resembles the timbre of the violin. The synthesized sound disputes the role of protagonist with the instrumental violin sound, thus its presence and absence (amplitude envelope) is in constant dynamic motion.

The frequency spectrum of the generated sound is constantly changing in a continuous slow oscillating motion between consonance and dissonance in the overall harmonic perception.

Sonority State 3: The electronic part performs similarly to sonority state 2, however, at this state it performs exaggerated modulations that transform not only the spectrum but also the timbre, pushing the synthesized sound to the limit where it sounds as a "microphone-feedback".

Additionally, all three sonority states should present a common behavior: in order to highlight the instrumental repetitiveness, the produced sound should evidence the repetitive frequencies with increasingly longer duration and amplitude than the other frequencies.

From the identified sonority states it was possible to observe that all of them presented similar morphological behavior. Their uniqueness is evidenced by the different sonic aesthetics and range or radicalism of their modulating parameters.

The strongest conceptual aspect that is evident in the instrumental writing and consequently also evident in the described sonority states is the repetitiveness of material and

its aesthetic consequences. Hence, the particle-based model should also be developed with the primordial objective of infusing the respective synthesis processes with this characteristic. However, the electronically generated sounds should not present repetitive or periodic sounds, rather, it should highlight this aspect by reacting to the instrumental repetitiveness and somehow empowering it.

In order to achieve the delineated objectives, the idea was to create a particle-based model inspired by Kepler's *Law of Planetary Motion*, which describes the orbit of planets around stars, and Newton's *Law of Universal Gravitation*. In this analogy, specific notes from the harmony would be given the role of stars surrounded by a big number of planets traveling freely around and among them.When a planet moves close to a star, it starts performing elliptical orbits around that corpus. Each planet represents a possible frequency of the electronic sound, and they are set with no initial frequency nor amplitude. Planets maintain this condition while traveling around the environment without engaging in an actual orbit motion until it is "trapped" by the gravitational field of a star. At which point, it assumes the harmonic role of that star by acquiring its note frequency or other harmonic partial of that frequency according to a probability distribution function.

From this model and analogy, the repetitiveness aspect could be enforced by controlling the mass of each star, and consequently its gravitational field (range of influence). Each star starts with an initial mass that is set to be the minimum value, each time the violin plays the note relative to that star, its mass is increased. Ergo, the more repetitions of a note, the bigger and heavier the respective star becomes, and the more frequencies related to that fundamental note will be present on the electronic sound. To counterbalance the increase in mass, each new iteration of the system, the mass of a each star is decreased, until it reaches the minimum value again. Thus, if the violin don't play the star's note, it will shrink down (or maintain) to the initial minimum size.

If on the behavioral perspective, the proposed model could be used to implement all identified sonority states, from the aesthetic perspective different types of audio processes needed to be designed in order to fulfill the delineated criteria.

For sonority state 1, where the electronic output should be smooth and restrictively consonant with the harmony of the instrumental part, the strategy was to create a simple additive synthesis, where each particle (planet) represented one sine wave oscillator, and only harmonic pertinent notes (stars) are present on the particle-system. The amplitude of each oscillator is controlled by the planets orbit radius and velocity and a function that dictates that the amplitude of an oscillator is zero, until it starts an orbit movement. Additionally, gaussian amplitude envelopes are used in order to prevent abrupt changes in the overall amplitude.

For sonority states 2 and 3, where the electronic output should present a timbre similar to that of the instrumental input, the strategy was to use a delay with feedback as the respective audio process for each planet. By setting the feedback value to values between 0.98 and 1., and setting the delay time with the frequency of its reciprocal planet, each planet becomes an instance of an audio process equivalent to a waveshaping synthesis (Roads, 1979) audio process.

A digital delay configured with a very high feedback value and a very small delay time (e.g. 440Hz represents a delay time of about 2.28ms) acts similarly to a table look up oscillator, which instead of using sinusoidal values, it uses the values stored in a generic buffer to generate the output signal. In this case, the buffer contains the incoming audio signal of the violin and the generated signal output is pitched by the delay time, but with the timbre shaped by the violin audio.

Besides the timbre possibilities, another upside of the use of delay lines with feedback to implement sonority states 2 and 3 is the different kinds of modulations that can be achieved just by controlling the delay time and feedback parameters. By mapping these values to the planet's orbit radius and velocity it is possible to achieve the rich glissandi and also the "microphone-feedback" sound desired by the composer.

The final particle-system is thus composed of three distinct types of particles: stars, additive synthesis planets, waveshaping synthesis planets. Their generic and specific attributes configuration is presented in table 5.3.

Particle type	Generic attributes	Specific attributes
Star	position mass velocity	frequency
Additive-synthesis planet		orbit status orbit radius frequency amplitude
Waveshaping-synthesis planet		orbit status orbit radius frequency (delay time) feedback amplitude

Table 5.3: Generic and specific particle attributes of the particle system implementation used in *Nomoi*

5.3.3.2 Software Implementation

The live-electronics system of Nomoi was entirely developed using the Max programming environment. However, while the audio processes were implemented using Max's graphical programming paradigm, the particle system was implemented in C/C++ with the Max Software Development Kit (Max SDK) and compiled to run as a native Max object, hence, the communication of the particle system and the respective audio process is taken care inside the application using common Max language messages.

Differently to the other presented cased studies (*O Farfalhar das Folhas* and *Changeless*, sections 5.1 and 5.2 respectively) the final application does not present a graphical visualization of the particle system. The system is completely transparent to the composer, it was used only as an implementation strategy by the musical assistant. Therefore, the graphical output is omitted from the final application to spare processing power.

The final Max application of the live-electronics system for *Nomoi* presented two main windows: the concert performance graphical user interface (GUI), and the event editor window.

The concert performance GUI (Figure 5.12) contains all the elements necessary for cal-

ibrating the microphones input and final audio output, the interface for triggering the global events that are precisely marked on the instrumental score, and a visualization of the spatialization.

The composer had designed very specific spatial trajectories for the electronic sounds, thus, the spatialization of the synthesized sound is not individualized per audio process instance (per particle), instead, the audio of all instances are mixed prior to the DBAP (distance based amplitude panning) spatialization algorithm.

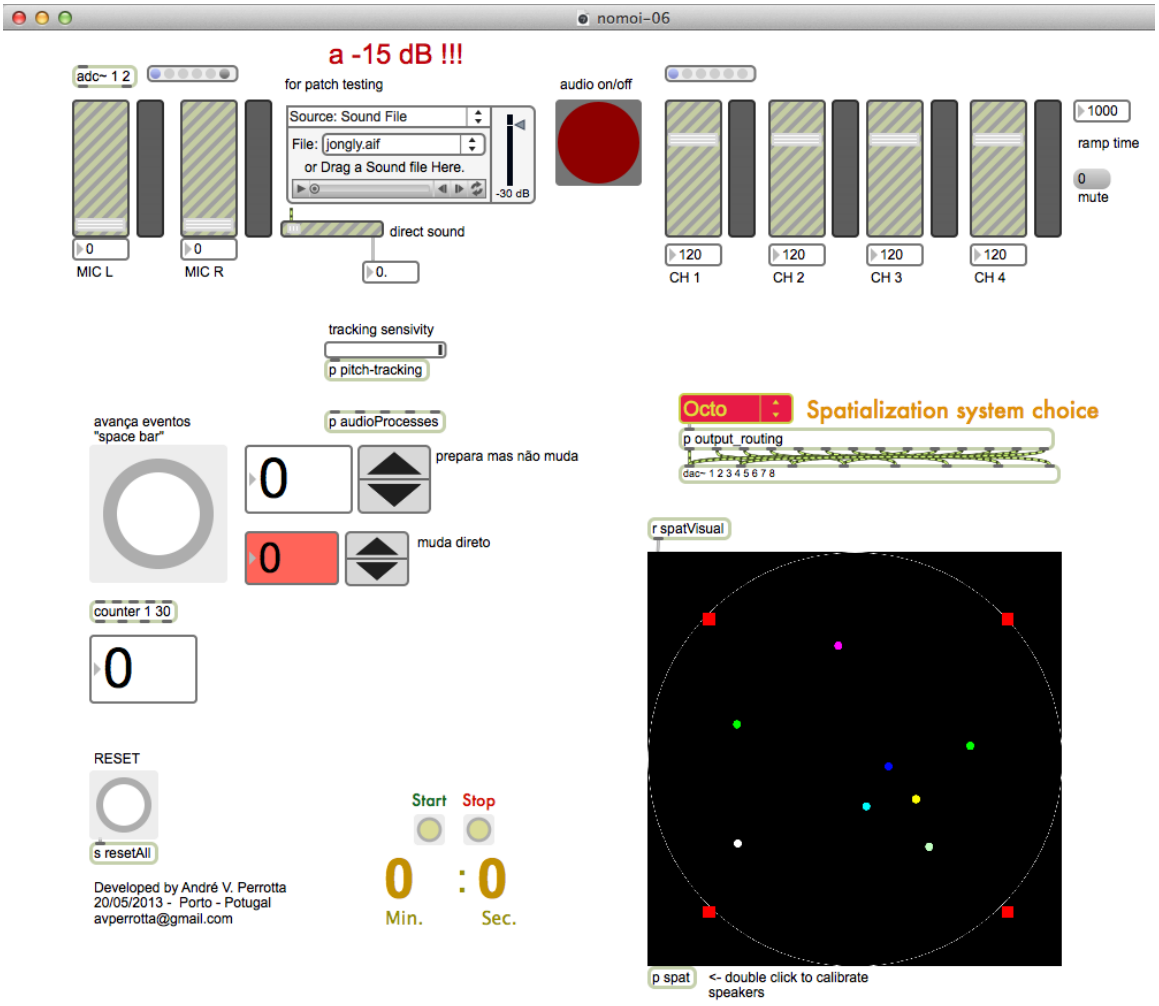


Fig. 5.12: Graphical user interface of the live-electronics Max application of *Nomoi*

The event editor window provided the composer with the possibility of configuring the parameters of the audio process of each event. In the case of the events that used the additive

or waveshaping synthesis controlled by the particle system, the composer could control a timbre parameter, which in turn would control the probability of injecting additive or waveshaping particles into the system, the range of the generated spectrum, and range of feedback and delay modulation (only valid for the waveshaping audio process).

The additive synthesis audio process instance is implemented with a very simple DSP chain containing a sinusoidal oscillator, and an amplitude envelope (Figure 5.13). The waveshaping audio process instance is implemented with an also simple delay line with feedback that receives and processes the audio input from the violin (Figure 5.14).

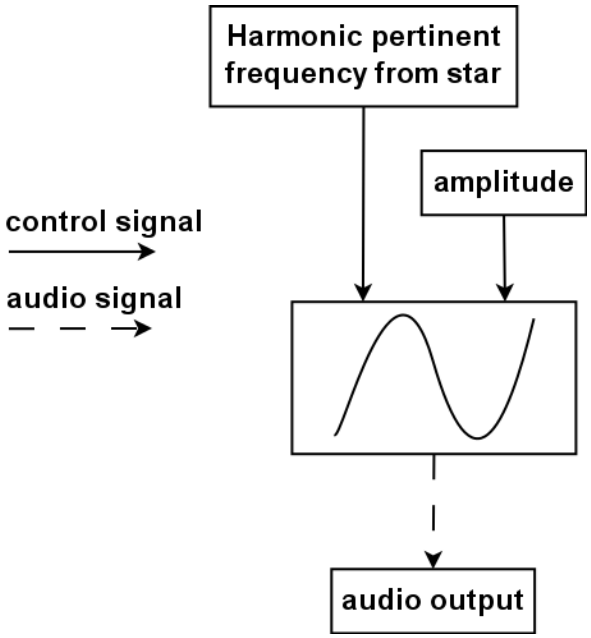


Fig. 5.13: Additive synthesis particle audio process used in *Nomoi*

5.3.4 Discussion and Composer Feedback

In *Nomoi* the particel system modelling technique provided a great contribution to the work developed by the musical assistant. the composer had a clear aesthetic and conceptual idea in his mind, that he aspired to "materialize" in the live-electronics. This clear sonic image, however, required a great deal of experimentation and heuristic strategies to be achieved.

In that matter, the particle-based model served as a strong basis for experimenting several different synthesis techniques. By having the conceptual behavior already delineated

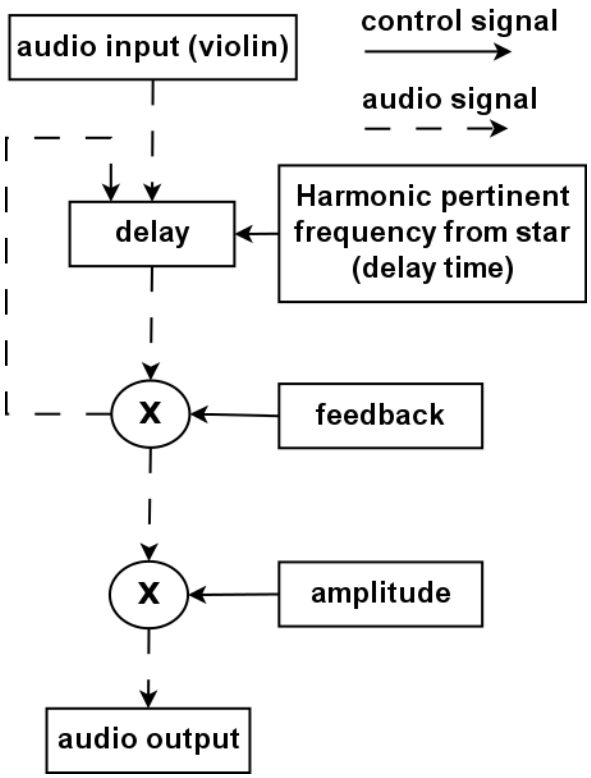


Fig. 5.14: Waveshaping synthesis particle audio process used in *Nomoi*

and implemented, the composer and musical assistant could concentrate on finding the best solution for the aesthetic criteria.

On a technical perspective, the use of the Max SDK to develop and implement the particle system proved to be of efficient in terms of computational power, specially because it runs inside Max using its native language for communication, and also in term of practicality, by eliminating the need of a secondary stand-alone application.

On the composer's perspective, the final result of the live-electronics system met his musical aspirations and expectations. He highlights the fact that the development process and methodology assisted in the experimentation process, and consequently, in finding the optimal solution for the proposed musical problems.

The composer also expresses that the possibility of editing events was very interesting, and that he took advantage of it and redistributed or reformulated some of the instrumental phrases and pauses duration and order to fit sometimes more and sometimes less electronic sounds.

Finally, he states that the implemented audio processes are very interesting and that he already composed another piece, *Nomoi II* (2014) for violin and live-electronics, that uses the same live-electronics system.

Chapter 6

Particle Systems Playground Toolkit

From the work developed in the case studies presented in chapter 5, it was possible to understand the practical and technical challenges that are involved in the usage of the particle system modelling method. The task of implementing the particle system from scratch using C/C++ programming language is only accessible or suitable for experienced programmers, otherwise the developer will spend much more time dealing with programming issues than to actual creative and musical problems.

Hence, it became clear that the next step in the research process was to develop a software framework and toolkit that could standardize and assist the technical implementation of live-electronics systems that take advantage of the proposed particle system modelling technique.

For that purpose the *Particle Systems Playground* (psPlayground) framework and toolkit was developed and implemented.

The psPlayground toolkit is publicly available at:
https://github.com/avperrotta/psPlayground

6.1 Description

The psPlayground toolkit is a collection of open source Max objects and abstractions developed in C/C++ using the Max SDK that are designed to assist the development of real-time

audio applications based on the proposed particle system modelling technique.

The psPlayground toolkit is designed for composers, musical assistants and any Max enthusiasts that have a basic understanding of the Max visual programming environment and also computer music techniques. It enables any developer to easily create Max patches that take advantage of the prominent features of the particle system modelling technique such as the per-audio-process-instance spatialization, layering and micromodulations and stochastic transitions (these features are discussed in section 4.3.4).

The toolkit is developed with an architecture that enables the user to create functional patches without the need of complex logical programming. A single abstraction named "psp.maxpat" takes care of creating, updating, and rendering the particle systems. The user controls everything just by sending and receiving messages to that abstraction. By instantiating the "psp.maxpat" abstraction with a specific (user generated) name, all other objects and abstractions of the toolkit that are instantiated with that same name are immediately attached to the context. Allowing the audio and control signals to flow on the system background without the user having to create manual connections between the objects and abstractions of the toolkit.

Figure 6.1 displays the simplicity of the usage of the toolkit. Only two objects and one message are necessary for creating, updating and rendering a rain simulation particle system.

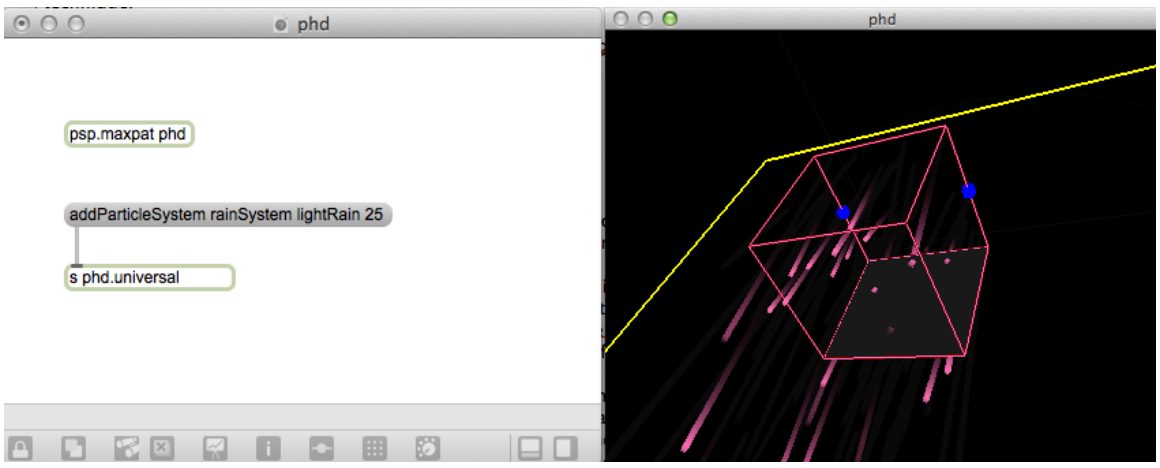


Fig. 6.1: psPlayground toolkit example. Only two objects and one message are necessary for creating, updating and rendering a rain simulation particle system

The toolkit main features are:

- Miscellanea of particle systems and particle types such as random movement system, rain system, particles injectors, attractors.
- Complete DBAP (distance based amplitude panning) engine with graphical user interface for calibrating room and algorithm parameters (Figures 6.2 and 6.3), trajectory proportional reverb and spectral delay features, no (software) limitation regarding the number of independent speaker channels, the ability to generate geometrical 3D trajectories using cartesian, spherical and cilindrical coordinates.
- Recording/playBack of particles trajectories.
- No (software) limit to the number of particles or the number of particle systems.

New particle systems and particle types can be programmed in C++ using the available psPlayground framework. The user can create new particle systems and particle types by "copying" and modifying the examples. In order to provide a simple programming strategy, the particle systems and particle types source code is organized using object-oriented methodology with an application loop routine similar to that of the OpenFrameworks and Processing programming frameworks, which in turn is based on the setup/update/draw routine. Where the setup function executes once when the program starts, and the update and draw routines are executed at each iteration of the system.

The toolkit provides a simple tutorial and examples that shows step by step how to use the objects and abstractions with the available particles and particle systems, how to create a particle system controlled audio synthesis with layering, micromodulations, stochastic transitions and particle/audio process independent spatialization with the DBAP spatialization engine.

6.2 Practical Applications

The psPlayground toolkit and framework is already being used in professional projects.

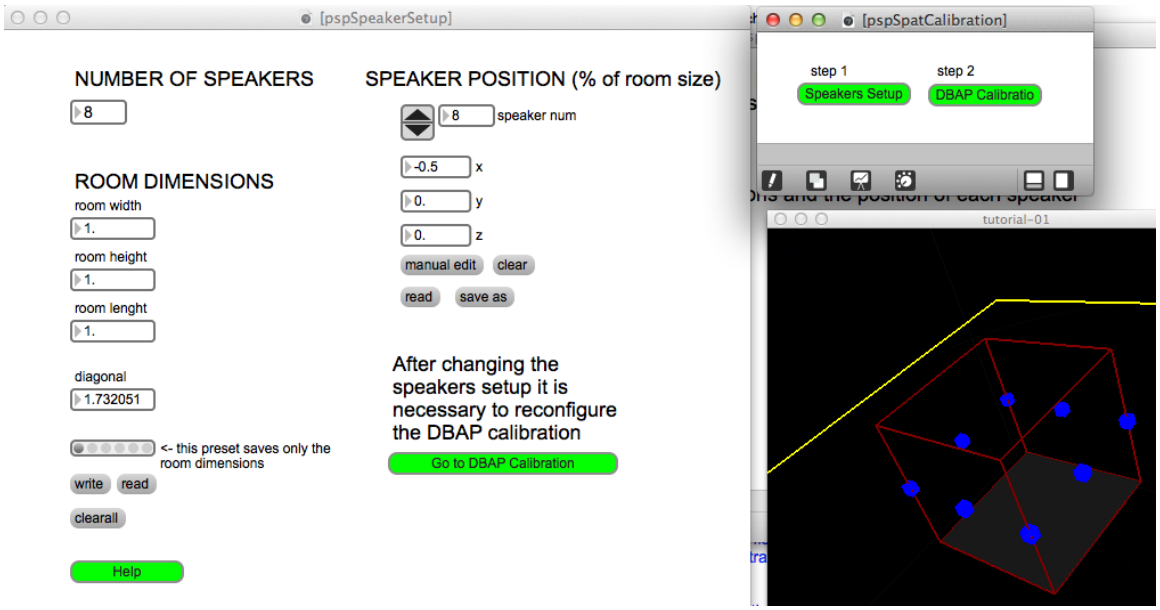


Fig. 6.2: psPlayground DBAP spatialization engine graphical user interface for calibrating the room dimensions, number and position of speakers

The psPlayground toolkit was used in the development of the live-electronics system of the mixed music work *Madrigal* by the composer Marcus Siqueira, for 11 string guitar and live-electronics. Dedicated to the virtuous guitar player Daniel Murray, this work is scheduled to be premiered in June/2015.

The toolkit is being used in the undergoing development of the new version of the MusicPanSPace Spatialization (MPSP) software, developed in collaboration with the composer Flo Menezes, in the PANaroma Electroacoustic Studio of the São Paulo State University (UNESP), in the scope of the FAPESP (São Paulo Research Foundation) funded project *Electroacoustic sound diffusion and spatialization in real time via MPSP* (Menezes, 2013).

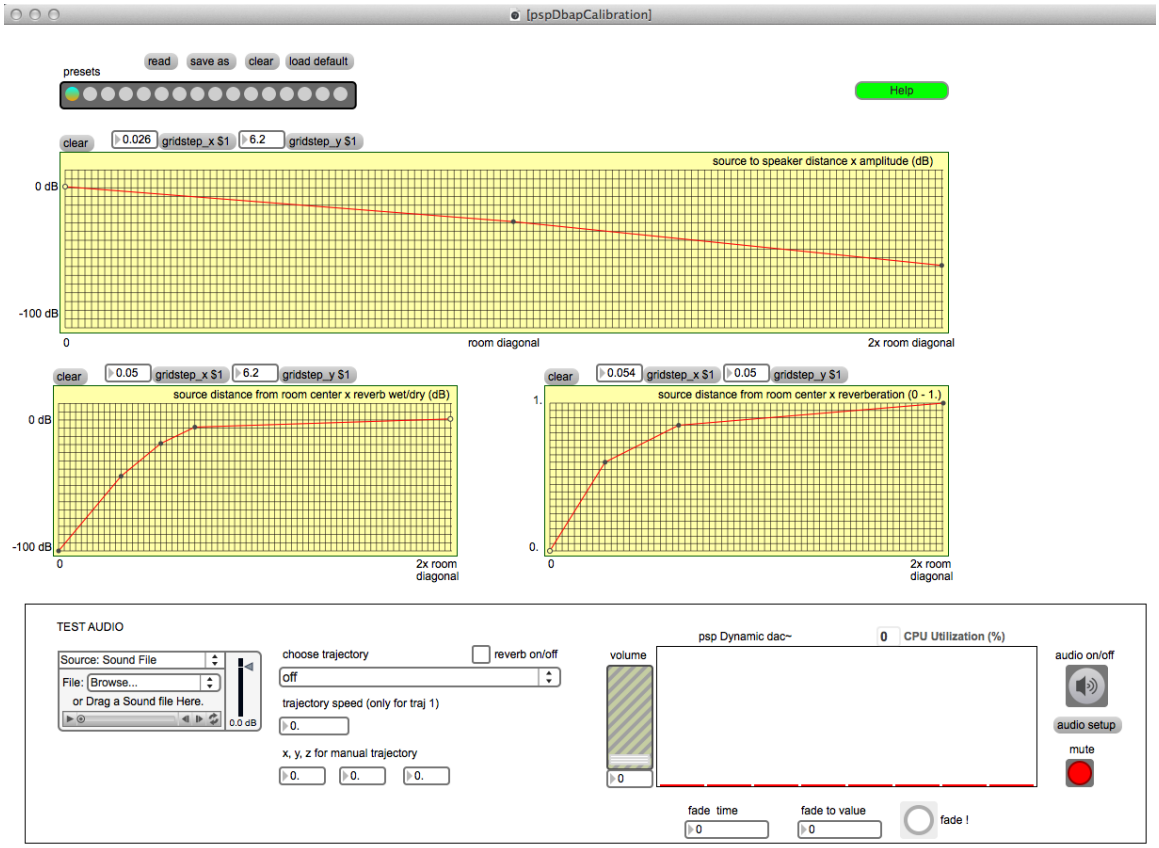


Fig. 6.3: psPlayground DBAP spatialization engine graphical user interface for calibrating the algorithm parameters

Chapter 7

Conclusion

This thesis presents a theoretical and practical framework that can be used in the elaboration of live-electronics systems of mixed music compositions.

The researched was centered around the fundamental question (presented in section 1.2):

> "Is it possible to use logical constructions and computational methods to mediate the process of translating abstract artistic ideas into executable software in mixed music works, using for that purpose an implementation method based on particle systems ?"

In order to answer this question a thorough identification and analysis of the technical, conceptual and practical aspects of the development, implementation and performance of live-electronics systems have been developed.

From the identification of these structures and processes a generic formalized logical representation of such systems was proposed. This became the theoretical basis for devising the theoretical framework for modelling the live-electronics systems using particle systems.

The formalized theoretical framework was put to test in the development of live-electronics systems for different compositions by different composers and also implemented in the form of software toolkit and framework.

7.1 Research Findings

The proposed particle systems modelling method proved to be effective in the task of assisting the conceptual translation of musical and aesthetic ideas into implementable audio processing software.

By acting as a bridge between the composer's abstract universe to the musical assistant's technical and computational universe, the particle systems modelling method thrived not only in dealing with the conceptual mapping problem, but also in the restrictively technical aspects of the implementation of live-electronics software such as the clear separation of global and local control structures and conceptual and parametric mapping layers.

The use of particle systems for modelling audio processing software also presented new features such as the per-audio-process spatialization possibilities, micromodulations, layering and stochastic transitions between different audio processes, that consequently expands the musical possibilities of electronic composition.

The proposed modelling strategy proved to be most effective in the development of live-electronics where the electronic part is either independent or isochronous to the instrumental part. In this situations, the audio processes that implement the electronic part rely on the real-time control of a significant number of variables. Controlling these variables by means of well designed mapping strategies within the particle systems allows the creation and control of complex dynamic sonic and morphologic behaviors by direct control over a small number of variables.

7.2 Contributions

The main contribution to knowledge presented in this research work is the theoretical formalized description of the structures and processes that implement a live-electronics system and the applied particle systems modelling technique to this formalization.

The development of such formal framework significantly contributes to the consolidation of the work of a musical assistant. In a context where artistic works are increasingly interdisciplinary and involves several technical and technological concerns, the work of a

musical assistant reciprocally requires a more structured and assertive development methodology in order to face the contemporary and state-of-art conceptual, artistic and aesthetic challenges.

Additionally to the theoretical framework, the implemented psPlayground software toolkit and framework represents a significant contribution by offering an entry point to the implementation and experimentation of the proposed particle systems modelling method for non-expert developers, composers and musical assistants.

7.3 Future Work

Throughout the course of the presented research the author dealt with different subjects and knowledge domains. The ones that were most fundamental and important to the delineated research objectives, such as the subjects related to computer music in the context of mixed music works and the live-electronics systems technicality, were thoroughly explored and thus culminated in the proposed method, case studies and conclusion. Other subjects were briefly explored, nevertheless they recurrently crossed the main research path.

In that matter, the interpersonal relationship between musical assistant and composer is a subject that could be deeper explored. Even though the collaborative work between "technician" and "artist" is *a priori* technical and artistic respectively, in practice these roles and boundaries are not clear nor exact. Moreover, this distinction between technical and artistic parts is also clouded in the final product, which rises some social problems related to authorship, acknowledgment and acceptance.

Research on this topic is fundamental for the establishment of the professional status of the musical assistant, and also for the role of technical/technological development in any artistic endeavor that lies on the interface of art and technology. This is a problematic common to all contemporary arts, and even though it has received some attention from the scientific community, and some important works such as (Poletti et al., 2002; Trifonova et al., 2009; Zattra, 2006, 2013) have been developed, it still far from being a broadly discussed and explored topic.

In what regards the developed particle system modelling method, the scope of the research and work presented on this thesis was limited to the mixed music compositions and more specifically on the realm of "serious" music. Widening the scope of the research to encompass other musical genres, composers, musicians and most importantly the contemporary laptop musicians (Cascone, 2003; Collins, 2003), would significantly add to the evolution of the method.

In the specific case of laptop music, given that the music composition concepts and performance context differs drastically from the research mixed music context, modifying and applying the proposed method to the requirements of this music genre could not only serve as a wider validation base, but also expand its technical potential and forms of usage.

In what regards the technical implementation of the proposed method one important research path that can be explored is the development of software tools and applications that can assist the use of the method in different audio development platforms such as the digital audio workstations. Wherefrom the method could be explored by sound designers in their studio practice.

Finally, it is the author's objective to develop and implement a stand-alone version of the psPlayground toolkit and framework to be used as a pure controller for multimedia applications, supported by usual desktop computers and also mobile devices in the form of stand-alone application or Audio Unit (AU) or VST plugin. This psPlayground application will present a comprehensive library of particle systems and particle types, with models for simulating physical, chemical, biological and social phenomena, which the user will be able to create and control through a simple user interface without the need to use any kind of programming language. The initial steps towards this objective has already been taken, and the initial code base of this application has already been implemented using the Juce C++ cross-platform library (Juce, 2015) and is publicly available at: https://github.com/avperrotta/psPlayground/tree/master/psPsa-001.

Bibliography

Banzi, M. (2009). *Getting Started with arduino*. " O'Reilly Media, Inc.".

Battenberg, E., Freed, A., and Wessel, D. (2010). *Advances in the parallelization of music and audio applications*. Ann Arbor, MI: MPublishing, University of Michigan Library.

Bisig, D. and Kocher, P. (2012). *Tools and Abstractions for Swarm based Music and Art*. Ann Arbor, MI: MPublishing, University of Michigan Library.

Bisig, D., Neukom, M., and Flury, J. (2008). Interactive swarm orchestra-a generic programming environment for swarm based computer music. In *Proceedings of the International Computer Music Conference. Belfast, Ireland*.

Blackwell, T. and Young, M. (2004). Swarm granulator. In *Applications of Evolutionary Computing*, pages 399–408. Springer.

Boulez, P. (1978). Technology and the composer. *Leonardo*, pages 59–62.

Bullock, J. (2008). *Implementing audio feature extraction in live electronic music*. PhD thesis, University of Birmingham.

Buxton, W., Reeves, W., Baecker, R., and Mezei, L. (1978). The use of hierarchy and instance in a data structure for computer music. *Computer Music Journal*, pages 10–20.

Cádiz, R. and Kendall, G. S. (2005). A particle-based fuzzy logic approach to sound synthesis. In *Proceedings of the Conference on Interdisciplinary Musicology, Montreal, Canada*.

Cage, J. (2011). *Silence: lectures and writings*. Wesleyan University Press.

Cascone, K. (2003). Grain, sequence, system: Three levels of reception in the performance of laptop music. *Contemporary Music Review*, 22(4):101–104.

Catto, E. (2010). Box2d.

Chadabe, J. (1997). Electric sound: the past and promise of electronic music.

Chadabe, J. (2002). The limitations of mapping as a structural descriptive in electronic instruments. In *Proceedings of the 2002 conference on New interfaces for musical expression*, pages 1–5. National University of Singapore.

Chion, M. (1988). Les deux espaces de la musique concrète. *L'espace du son*, pages 31–3.

Chowning, J. M. (1977). The synthesis of complex audio spectra by means of frequency modulation. *Computer Music Journal*, pages 46–54.

Collins, N. (2003). Generative music and laptop performance. *Contemporary Music Review*, 22(4):67–79.

Coumans, E. (2006). Bullet physics library.

De Poli, G. and Rocchesso, D. (1998). Physically based sound modelling. *Organised Sound*, 3(01):61–76.

DeLoache, J. S. (2005). Mindful of symbols. *Scientific American*, 293(2):72–77.

Fellgett, P. (1975). Ambisonics part one: General system description. *Studio Sound*, 17(8):20–22.

Fonseca, N. (2013). 3D Particle Systems for Audio Applications. *dafx13.nuim.ie*, pages 2–7.

Frengel, M. (2010). A multidimensional approach to relationships between live and non-live sound sources in mixed works. *Organised Sound*, 15(2):96–106.

Frigg, R. and Hartmann, S. (2009). Models in science. *Stanford encyclopedia of philosophy*.

Gerzso, A. (1984). Reflections on Répons. *Contemporary Music Review*, 1(1):23–34.

Haseman, B. (2006). A manifesto for performative research. *Media International Australia, Incorporating Culture & Policy*, (118):98.

Hockney, R. W. and Eastwood, J. W. (2010). *Computer simulation using particles*. CRC Press.

Hoffman, M. and Cook, P. R. (2007). Real-time feature-based synthesis for live musical performance. In *Proceedings of the 7th international conference on New interfaces for musical expression*, pages 309–312. ACM.

Hoffmann, P. (2002). Towards an" automated art": Algorithmic processes in xenakis' compositions. *Contemporary Music Review*, 21(2-3):121–131.

Honing, H. (1993). Issues on the representation of time and structure in music. *Contemporary music review*, 9(1-2):221–238.

Hunt, A. and Wanderley, M. M. (2002). Mapping performer parameters to synthesis engines. *Organised Sound*, 7(2):97–108.

Jehan, T. and Schoner, B. (2002). An audio-driven perceptually meaningful timbre synthesizer. *Analysis*, 2(3):4.

Juce (2015). Juce c++ cross-platform library. http://www.juce.com/.

Karaboga, D. and Akay, B. (2009). A survey: algorithms simulating bee swarm intelligence. *Artificial Intelligence Review*, 31(1-4):61–85.

Kaye, N. (2013). *Site-specific art: performance, place and documentation*. Routledge.

Kennedy, M. (1994). *The Oxford dictionary of music*. Oxford University Press, New York.

Kim-Boyle, D. (2005). Sound spatialization with particle systems. In *Proceedings of the 8th international conference on digital audio effects (DAFX-05), Madrid, Spain*, pages 65–68.

Kindler, E. and Krivy, I. (2011). Object-oriented simulation of systems with sophisticated control. *International Journal of General Systems*, 40(3):313–343.

Kipfer, P., Segal, M., and Westermann, R. (2004). Uberflow: a gpu-based particle engine. In *Proceedings of the ACM SIGGRAPH/EUROGRAPHICS conference on Graphics hardware*, pages 115–122. ACM.

Kostadinov, D., Reiss, J. D., and Mladenov, V. (2010). Evaluation of distance based amplitude panning for spatial audio. In *ICASSP*, pages 285–288.

Kovecses, Z. (2002). *Metaphor: A practical introduction*. Oxford University Press.

Lazzarini, V. (2004). Computer instrument development and the composition process. In *Proceedings of the 7th International Conference on Digital Audio Effects (DAFx04), Naples, Italy, October 5*, volume 8.

Lee, M. and Wessel, D. (1992). Connectionist models for real-time control of synthesis and compositional algorithms. In *Proceedings of the International Computer Music Conference*.

Leman, M. (2002). Expressing coherence of musical perception in formal logic. In *Mathematics and Music*, pages 185–198. Springer.

Lévi-Strauss, C. (1966). *The savage mind*. University of Chicago Press.

Lossius, T., de la Hogue, T., Baltazar, P., Place, T., Wolek, N., and Rabin, J. (2014). Model-view-controller separation in max using jamoma. In *Proc. of the joint ICMC SMC 2014 Conference, Athens, Greece*.

Malham, D. G. (1998). Approaches to spatialisation. *Organised sound*, 3(02):167–177.

Manoury, P. (1998). Les partitions virtuelles, in: P. manoury, la note et le son.

May, A. (1999). Philippe manoury: Jupiter [review article]. *Computer Music Journal*, 23(3):103–104.

McCartney, J. (1996). Supercollider: a new real time synthesis language.

McCartney, J. (2002). Rethinking the computer music language: Supercollider. *Computer Music Journal*, 26(4):61–68.

Menezes, F. (1997). To be and not to be: aspects of the interaction between instrumental and electronic compositional methods. *Leonardo Music Journal*, pages 3–10.

Menezes, F. (2002). For a morphology of interaction. *Organised Sound*, 7(3):305–311.

Menezes, F. (2004). *A acústica musical em palavras e sons*. Atelié Editorial.

Menezes, F. (2006). *Música maximalista: ensaios sobre a música radical e especulativa*. UNESP.

Menezes, F. (2013). Electroacoustic sound diffusion and spatialization in real time via MPSP. FAPESP Regular Grants, Instituto de Artes (IA). Universidade Estadual Paulista (UNESP).

Miranda, E. R. (1995). Granular synthesis of sounds by means of a cellular automaton. *Leonardo*, pages 297–300.

Montague, S. (1991). Live electronics - introduction. *Contemporary Music Review*, 6(1):85–88.

Morgan, R. P. (1991). *Twentieth-century music: a history of musical style in modern Europe and America*. Norton New York.

N. Schnell, D. S. (2005). Gabor, multi-representation real-time analysis/synthesis. *Proc. of the 8th Int. Conference on Digital Audio Effects (DAFx'05),*, pages 1–5.

Nicolas, F. (2002). Questions of logic: Writing, dialectics and musical strategies. In *Mathematics and Music*, pages 89–111. Springer.

Nort, D. V., Wanderley, M., and Depalle, P. (2014). Mapping control structures for sound synthesis : Functional and topological perspectives. *Computer Music Journal*, 38(4).

Orio, N., Lemouton, S., and Schwarz, D. (2003). Score following: State of the art and new developments. In *Proceedings of the 2003 conference on New interfaces for musical expression*, pages 36–41. National University of Singapore.

Peignot, J. (1960). De la musique concrète à l'acousmatique. *Esprit*, (28):111–120.

Penha, R. (2014). *Modelos de espacialização: integração no pensamento composicional*. Phd thesis, Universidade de Aveiro.

Piskunov, N. S. and Yankovsky, G. (1974). *Differential and integral calculus*, volume 1. Mir Moscow.

Poletti, M., Mays, T., and Faia, C. (2002). Assistant musical ou producteur ? Esquisse d'un nouveau métier. In *Journées d'Informatique Musicale*.

Puckette, M. (1997). Pure data: Recent progress. In *Proceedings of the Third Intercollege Computer Music Festival*, pages 1–4. Citeseer.

Puckette, M. (2002). Max at seventeen. *Computer Music Journal*, 26(4):31–43.

Puckette, M. (2006). *Theory and techniques of electronic music*. World Scientific.

Puckette, M. and Lippe, C. (1992). Score following in practice. In *Proceedings of the International Computer Music Conference*, pages 182–182. INTERNATIONAL COMPUTER MUSIC ACCOCIATION.

Pulkki, V. (1997). Virtual sound source positioning using vector base amplitude panning. *Journal of the Audio Engineering Society*, 45(6):456–466.

Reenskaug, T. (1979). Models-views-controllers. *Technical note, Xerox PARC*, 32:55.

Reeves, W. T. (1983). Particle systems—a technique for modeling a class of fuzzy objects. In *ACM SIGGRAPH Computer Graphics*, volume 17, pages 359–375. ACM.

Rentsch, T. (1982). Object oriented programming. *ACM Sigplan Notices*, 17(9):51–57.

Reynolds, C. W. (1987). Flocks, herds and schools: A distributed behavioral model. *ACM SIGGRAPH Computer Graphics*, 21(4):25–34.

Risset, J.-C. (1978). *The development of digital techniques: a turning point for electronic music?* Rapport Ircam 9/78, 1978, Centre Georges Pompidou.

Risset, J.-C. (1989). Paradoxical sounds. In *Current directions in computer music research*, pages 149–158. MIT Press.

Risset, J.-C. (1999). Composing in real-time? *Contemporary Music Review*, 18(3):31–39.

Roads, C. (1979). A tutorial on non-linear distortion or waveshaping synthesis. *Computer Music Journal*, pages 29–34.

Roads, C. (1996). *The computer music tutorial*. MIT press.

Roads, C. (2004). *Microsound*. The MIT Press.

Roads, C. and Strawn, J. (1985). *Foundations of computer music*. Massachusetts Institute of Technology.

Rothstein, J. (1992). Midi: A comprehensive introduction. ar editions. *Inc., Madison, WI*.

Rowe, R. (1999). The aesthetics of interactive music systems. *Contemporary Music Review*, 18(3):83–87.

Sadie, S. E. (1980). The new grove dictionary of music and musicians.

Scaletti, C. (2002). Computer music languages, kyma, and the future. *Computer Music Journal*, 26(4):69–82.

Schacher, J. C., Bisig, D., and Kocher, P. (2014). The map and the flock: Emergence in mapping with swarm algorithms. *Computer Music Journal*, 38(3):49–63.

Smith, J. O. (1992). Physical modeling using digital waveguides. *Computer music journal*, pages 74–91.

Smith, R. (2006). Open dynamics engine (ode).

Solomos, M. (2006). The granular connection (xenakis, vaggione, di scipio...). In *Symposium The Creative and Scientific Legacies of Iannis Xenakis International Symposium*.

Stam, J. (2003). Real-time fluid dynamics for games. In *Proceedings of the game developer conference*, volume 18, page 25.

Stroppa, M. (1999). Live electronics or... live music? towards a critique of interaction. *Contemporary Music Review*, 18(3):41–77.

Sturm, B. L. (2000). Sonification of particle systems via de broglie's hypothesis. In *International Conference on Auditory Display, Atlanta, GA*.

Taruskin, R. (2009). *Music in the late twentieth century: The Oxford History of Western Music*. Oxford University Press.

Tindale, A. R., Kapur, A., Tzanetakis, G., Driessen, P., and Schloss, A. (2005). A comparison of sensor strategies for capturing percussive gestures. In *Proceedings of the 2005 conference on New interfaces for musical expression*, pages 200–203. National University of Singapore.

Tremblay, P. A. (2006). Pragmatic considerations in mixed music: A case study of la rage. In *Proceedings of the International Computer Music Conference*, pages 527–530. The International Computer Music Association.

Trifonova, A., Ahmed, S. U., and Jaccheri, L. (2009). Sart: Towards innovation at the intersection of software engineering and art. In *Information Systems Development*, pages 809–827. Springer.

Vaggione, H. (1991). A note on object-based composition.

Viers, R. (2011). *The sound effects bible: how to create and record hollywood style sound effects*. Michael Wiese Productions.

Vorms, M. (2011). Representing with imaginary models: Formats matter. *Studies In History and Philosophy of Science Part A*, 42(2):287–295.

Wang, G. and Cook, P. R. (2008). *The chuck audio programming language. a strongly-timed and on-the-fly environ/mentality*. Princeton University.

Wright, M., Freed, A., and Momeni, A. (2003). Opensound control: State of the art 2003. In *Proceedings of the 2003 conference on New interfaces for musical expression*, pages 153–160. National University of Singapore.

Xenakis, I. (1960). Elements of stochastic music. *Gravesaner Blätter*, 18(1960):84–105.

Xenakis, I. (1992). *Formalized music: thought and mathematics in composition*. Number 6. Pendragon Press.

Zadel, M. and Scavone, G. (2006). Laptop performance: Techniques, tools, and a new interface design. In *Proceedings of the International Computer Music Conference*, pages 643–648.

Zattra, L. (2006). The identity of the work: agents and processes of electroacoustic music. *Organised Sound*, 11(02):113–118.

Zattra, L. (2013). Les origines du nom de rim (réalisateur en informatique musicale). In *Journées d'Informatique Musicale (JIM 2013), Saint-Denis*, pages 113–120.

Appendix A

Music compositions

1. Quaderno (2005, 2007) by Flo Menezes, for Marimba (2005) / Guitar (2007) and live-electronics, commissioned by Duo Contexto (Eduardo Leandro and Ricardo Bologna), first performed on June 16, 2006, in São Paulo, by Ricardo Bologna (marimba) and Flo Menezes (live-electronics), guitar version first performed on August 16, 2008, by Daniel Murray (guitar), Flo Menezes (live-electronics). Live-electronics software developed by Flo Menezes and André Perrotta.

2. *Madrigal* (2014) by Marcus Siqueira, for 11 string guitar and live-electronics, commissioned (and dedicated to) by Daniel Murray, live-electronics system software developed by André Perrotta, first performed in November 2014, in São Paulo - Brazil, by Daniel Murray (guitar)

3. *Crase* (2006) by Flo Menezes, for large orchestra and live-electronics, commissioned by the OSESP (Orquestra Sinfônica do Estado de São Paulo), first performed on December 13, 2007, in São Paulo by the OSESP, conducted by Victor Hugo Toro with Flo Menezes and André Perrotta controlling the live-electronics.

4. *O Farfalhar das Folhas* (2010) by Flo Menezes, 1 flutist (flute in C, in G and piccolo), 1 clarinetist (clarinet in Bb, in Eb and bass clarinet in Bb), violin, violoncello, piano and live-electronics, commissioned by MISO Music Portugal, first performed on July 3, 2010, in Lisbon, by the Sond'ArTe Electric Ensemble conducted by Jean-Sébastien

Béreau, with Flo Menezes, Paula Azguime and André Perrotta controlling the live-electronics.

5. *A Laugh to Cry* (2013) by Miguel Azguime, for 3 singers (2 sopranos and bass-baritone), 2 narrators, 7 instruments, live-electronics and video, commissioned by the Warsaw Autumn International Festival of Contemporary Music with the support of the Ernst von Siemens Music Foundation, first performed in September 27, in Warsaw, by the Norbotten NEO ensemble conducted by Petter Sundkvist, with technology direction by André Perrotta.